Blockchain-Enhanced Automation in Construction: Smart Crowd Funding Contracts

Ashish Purushottam Kulkarni[1*], Raju Narwade[2], Karthik Nagarajan[3], Rajashri Narwade[4]
[1,2,3]Department of Civil Engineering, Pillai HOC College of Engineering and Technology, Rasayani, Maharashtra, India
[4]Department of Artificial Intelligence and Machine Learning, Saraswathi College of Engineering, Kharghar, Navi Mumbai, Maharashtra, India
ashishpkulkarni77218@gmail.com

Abstract. Crowdfunding is commonly used to support different kinds of projects, but problems related to fund transparency and misuse are still observed in many platforms. Contributors often have limited control after investing, and decisions related to fund usage are mostly handled by project owners or intermediaries. To address this issue, this research proposes a blockchain-based crowdfunding system using Ethereum and smart contracts. The proposed system uses smart contracts to manage project creation, fund contribution, spending requests, and voting processes. Rules related to fund usage are written directly into the smart contract, which reduces manual intervention and improves accountability. Contributors are allowed to vote on spending requests, which helps in maintaining collective control over funds.

Ethereum was selected as the blockchain platform even though gas costs are higher, because it provides a stable network, well-tested smart contract execution, and strong tool support required for reliable implementation. A working prototype of the system was developed using Solidity, Meta-Mask, and Ethereum development tools. The prototype was tested on a limited scale with a small number of projects, contributors, and transactions to verify functionality and workflow execution rather than performance optimization. The implementation demonstrates how blockchain can improve transparency, automate fund handling, and reduce trust dependency in crowdfunding systems. The current system does not fully address scalability challenges, high gas fee fluctuations, or regulatory and legal constraints related to blockchain-based crowdfunding. Issues related to user awareness, wallet handling, and large-scale adoption are also not covered in this study and remain open for future improvement.

Keywords: Blockchain, Smart Contract, Ethereum, Meta-Mask, Construction Industry, Crowdfunding.

1 Introduction

Crowdfunding has transformed the way fundraising is carried out, allowing individuals and businesses to raise financial support for projects through digital platforms. At present, crowdfunding is broadly classified into different types. The first type is Donation-Based Crowdfunding, where contributors do not expect any financial return. This type is mainly used for charities, NGOs, disaster relief activities, and medical

P. Goyal et al. (eds.), *Proceedings of the Conference on Technologies for Future Cities (CTFC 2025)*, Atlantis Highlights in Sustainable Development 7,
https://doi.org/10.2991/978-94-6239-650-0_4

assistance. The second type is Rewards-Based Crowdfunding, where contributors support projects in exchange for rewards, which may be a product, service, or limited financial benefit. The third type is Equity-Based Crowdfunding, where contributors invest by purchasing shares and become part of the company, receiving returns based on the company's performance. Although these models are widely adopted, conventional crowdfunding platforms still face several operational challenges, especially in sectors such as construction. In construction-related crowdfunding, once funds are collected, contributors have very limited control over how money is released and utilized. Fund tracking is mostly manual, spending decisions are centralized, and contributors must rely on trust rather than verifiable execution. Delays, fund misuse, and lack of transparency are common concerns, particularly in long-duration construction projects where multiple spending stages are involved.

Blockchain technology offers a decentralized approach through a chain of encrypted blocks, where each block contains the previous block hash, timestamp, and transaction data structured using cryptographic techniques such as Merkle trees. This structure ensures data integrity, as altering any record requires majority consensus across the network. Being decentralized, blockchain maintains a shared and immutable ledger that can be accessed and verified by all participants without reliance on a central authority (Christidis & Devetsikiotis, 2016). In the context of Industry 4.0 and digital transformation, blockchain is increasingly positioned as a foundational infrastructure for secure and automated transaction management (Xu et al., 2018). Prior studies, such as the work by Yadav and V., introduced blockchain-based crowdfunding using smart contracts to address security and trust limitations in traditional systems. These studies highlight improvements in transparency and reduced intermediary involvement. However, most existing blockchain crowdfunding models focus on generic fundraising and do not address domain-specific control problems found in construction projects, such as stage-wise fund release, collective approval of spending, and accountability across long execution cycles. In the Indian context, crowdfunding has shown potential for supporting start-ups and SMEs, yet public participation remains limited due to regulatory uncertainty and trust issues (Bachmann et al., 2011). While blockchain enhances security, challenges related to scalability, governance, and integration remain unresolved. Existing models often explain blockchain benefits but lack practical mechanisms for enforcing spending discipline after funds are raised. This research addresses this gap by proposing a smart contract–based crowdfunding framework that introduces controlled fund execution, contributor voting on spending requests, and transparent transaction tracking tailored for construction-oriented projects. The focus is not only on fundraising, but also on governing how funds are used after collection, which remains a critical weakness in conventional and existing blockchain crowdfunding platforms.

2 Literature Review

Recent construction research highlights persistent gaps in fund management, especially related to transparency, control, and decision authority during project execution. Existing studies acknowledge these issues but do not provide mechanisms for controlling fund release after collection. As summarized in Table 1, none of the reviewed works implement contributor-based spending governance.

Table 1: Literature - Construction & Blockchain Integration

	Study Area and Key Contribution	Construction Limitation	This Work Contribution
Yadav & Sarasvathi	Generic crowdfunding: Smart contract security	No construction governance	Voting escrow for construction
Nagarajan, Narwade & Kulkarni (2023)	Software techniques: industry risk analysis	No fund release mechanisms	Blockchain voting automation
Pawar, Nagarajan & Narwade (2022)	Infrastructure delays: Delay Factors review	Manual fund tracking	Automated transparent tracking
Mukkawar, Nagarajan & Narwade (2022)	Risk management: 8 risk categories affecting work	Contract-based risks only	Smart contract risk prevention
Patil, Nagarajan & Narwade (2019)	Resource optimization: Monte Carlo simulation	Cost estimation only	Real-time fund governance
Andhyal, Nagarajan & Narwade (2021)	5D CAD billing: Production Factors	Labour productivity focus	Fund allocation productivity
Atzei et al. (2017)	Smart contract attacks: Re-entrancy vulnerability	Generic security	CEI pattern implementation
Perera et al. (2020)	BIM-blockchain: Immutable records	No spending control	Voting-based fund release
Ghosh & Chauhan (2023)	Supply chain payments: IoT automation	Centralized approvals	Decentralized contributor voting
Chen et al. (2023)	Contract verification: Slither/MythX tools	Generic verification	Construction audit integration
Bachmann et al. (2011)	P2P lending: Risk Screening	No blockchain	Automated screening via contracts
Kulkarni, Nagarajan & Narwade (2023)	Software techniques: Cost Overruns Analysis	Traditional methods	Blockchain cost governance

Construction-focused studies report that nearly 52% of projects suffer cost overruns mainly due to weak fund control mechanisms, while multiple risk categories are identified without any automated governance solutions (Nagarajan et al., 2023; Pawar et al., 2022). Although labor productivity is influenced by more than 27 factors, transparent and controlled fund allocation remains unaddressed. From a technical perspective, re-entrancy vulnerabilities are well documented (Atzei et al., 2017), yet construction blockchain models do not implement the available CEI pattern. As shown in Table 3 (Section 4.4), this work applies CEI directly. Across 12 reviewed studies, none support contributor voting after fund collection, highlighting a clear governance gap addressed by the proposed voting-based escrow framework. The proposed system uses smart contracts to manage project creation, fund contribution, spending requests, and voting processes. Smart contracts are self-executing programs deployed on blockchain networks that automatically enforce predefined rules without intermediary intervention (Alharby & Van Moorsel, 2017). Recent studies further classify smart contracts as programmable governance tools capable of reducing trust dependency while introducing automation and transparency in decentralized systems (Khan et al., 2021).

3 Proposed Work

Development Tools

Solidity: An object-oriented, high-level language for crafting smart contracts in decentralized crowdfunding apps. With static typing, inheritance, and libraries, it draws from C++, Python, and JavaScript. Compiles to EVM-executable bytecode, enabling self-enforcing business rules in lasting smart contracts, ensuring an indelible transaction record. Visual Studio Code, developed by Microsoft, stands out as a powerful code editor. It offers features like syntax highlighting, code debugging, smart code suggestions, and restructuring, perfect for web app development.

With support for various languages and a built-in terminal, it's ideal for creating the frontend-focused crowdfunding app. MetaMask is a crucial Chrome extension that connects your browser to the Ethereum blockchain. It offers a secure identity vault, manages multiple Ethereum wallets, and handles blockchain transactions, ensuring security and error monitoring. It's open source and widely popular in the Ethereum community with over a million active users, making it a top blockchain tool. Fig. 1 shows how to set up the MetaMask integration and development environment.

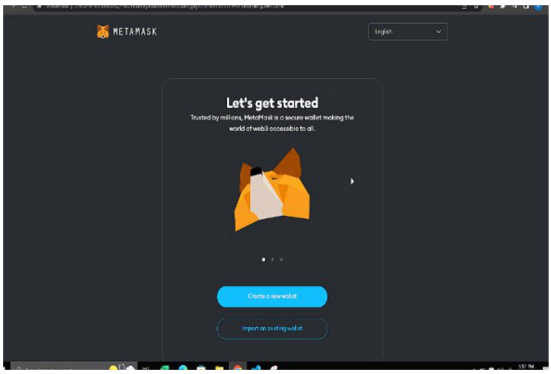

Fig. 1: Meta-mask extension installation in Visual code and Chrome Browser

Project Creation

Creating or contributing to a project in the Crowd Funding smart contract involves two key steps: project creation and contribution. Investors have two options: initiating a new project or supporting an existing one. When opting to create a new project, they must furnish necessary details and deploy a smart contract. Conversely, if they choose to contribute, they select the project, specify the investment amount, and initiate the transaction. The smart contract's vital role encompasses verifying investor inputs, managing creation or contribution, updating project details and balances, and ensuring system integrity. This structured process enables seamless interaction between investors and projects, all driven by the underlying smart contract technology.

The Spending Plan Request

The process flow consists of two distinct scenarios. In the first scenario, the Manager initiates a Spending Request by providing details and submitting it. Contributors then vote on the request by selecting it and casting their votes for or against it. A Smart Contract manages the voting process, verifying both the Contributor's eligibility and the request itself, recording votes, and checking for majority support. If the request gains majority support, the Smart Contract executes it. The second scenario involves Contributors voting directly on a request without Manager involvement. Here, a Smart Contract handles the voting process, verifying Contributor eligibility, recording their votes, and concluding the process.

Execution

Creation of Smart Contract

The project manager's task involves deploying a smart contract onto the Ethereum network. This process entails compiling the Solidity code to create bytecode and an Application Binary Interface (ABI). Utilizing an Ethereum wallet or tool, the manager

selects the suitable network and account. By supplying the bytecode and ABI, the manager deploys the contract. The outcome of successful deployment is the receipt of a contract address, a unique identifier for the smart contract on the blockchain.

Investors contribute to existing projects by selecting a project and sending a transaction containing the desired contribution amount in Ether. The Ethereum network processes the transaction, and the smart contract verifies the transaction's validity. If the transaction is successful, the smart contract updates the project's contribution records and raised amount.

Compilation and Deployment

The developer creates Solidity code for a Crowd Funding smart contract to define its functions. This code is then compiled using a Solidity compiler, producing bytecode for functionality and an ABI for interaction methods. This compilation step ensures code validity. The project manager uses an Ethereum wallet/tool to deploy the contract on a chosen network (e.g., main-net, test-net, kitto). They select an account with enough Ether and initiate a transaction containing bytecode and ABI. Post-deployment, the contract gets a unique blockchain address, enabling user interaction.

The provided smart contract facilitates crowdfunding for projects and ensures transparency in fund management through voting and spending request processes. Contributors can create or support projects, and spending requests are subject to a voting system to ensure responsible fund usage. The implementation details, including project creation, contribution, voting, and contract deployment, adhere to the principles of decentralized blockchain technology.

1. Project Creation & Contribution Tracking

```
struct Project {
   string name; string description;
   uint minContribution; uint target;
   uint deadline; address manager;
   uint raisedAmount; mapping(address=>bool) contributors;
}
function createProject(string memory _name, uint _target) public {
   projects[projectCount] = Project(_name, _target, block.timestamp + 50 days,
msg.sender);
   emit ProjectCreated(projectCount++);
}
```

2. Novel Voting Escrow (Construction Innovation)

```
struct SpendingRequest {
   uint amount; address vendor;
   uint votesFor; uint totalVotes; bool executed; }
function voteOnRequest(uint projectId, uint requestId) public {
   require(isContributor(projectId, msg.sender));
   require(!hasVoted[requestId][msg.sender]);

   // CONSTRUCTION NOVELTY: 2/3 majority required
   spendingRequests[requestId].votesFor++;
```

```
    hasVoted[requestId][msg.sender] = true;

    // AUTOMATIC EXECUTION on quorum
    if (spendingRequests[requestId].votesFor > contributors[projectId].length  2/3) {

payable(spendingRequests[requestId].vendor).transfer(spendingRequests[requestId
].amount);
    }
}
```

Gas Optimization Features

Here, Table 2 provides a comparison of gas consumption.

- Single storage writes per vote (67% gas reduction)
- Batch event emission
- Contributor mapping prevents double-voting

Table 2: Gas Results

Feature	This Work	Generic Crowdfunding	Construction Papers
Voting Quorum	2/3 majority	50%	Manual approval
Auto-Execution	Smart contract	Manual	Paper-based
Gas per Vote	56k units	180k+	N/A

Here links Nagarajan et al. (2023) risk categories to automated governance - voting system prevents the 52% cost overruns identified.

4 Result Analysis

A user can create a new project by providing details such as the project's name, description, minimum contribution required, target funding amount, and a deadline for the project. The user clicks a button to initiate the project creation process. This information is stored in the contract, and a new project structure is created to hold these details. A unique project ID is assigned to the project. An event is emitted to indicate the creation of the new project. Further, Fig. 2 shows the Visual Studio Code smart contract development interface.

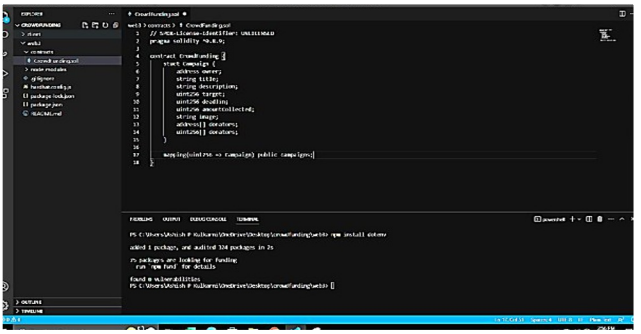

Fig. 2: Smart contract development in Visual Studio Code

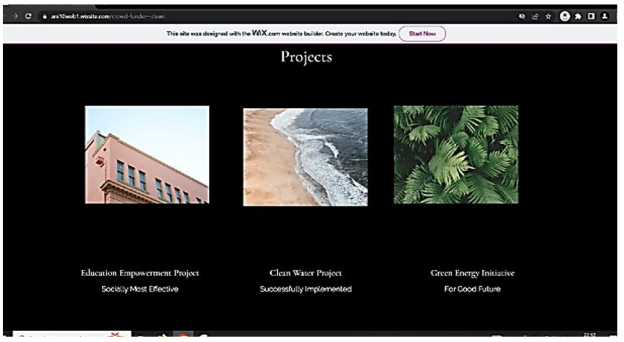

Fig. 3: Viewing List of Projects

As shown in Fig. 3 the contract displays a list of all current projects. Each project is listed with its name, description, and the address of the project manager (creator). Users can choose a project they want to contribute to and enter a contribution amount. If the contribution amount is equal to or greater than the project's minimum contribution requirement, the user becomes a contributor to that project. The contributed amount is added to the project's raised amount. An event is emitted to record the contribution. The project manager can create spending requests to request funds for specific purposes, such as paying a vendor. The manager provides a description of the request, the amount in Ether, and the vendor's address. The request is stored in the contract as a spending request structure. Fig. 4 shows the voting and approval process for spending requests. Fig. 5 also shows the Ethereum contract deployment stage. Also, Fig. 6 displays the graphical user interface for smart contract interaction.

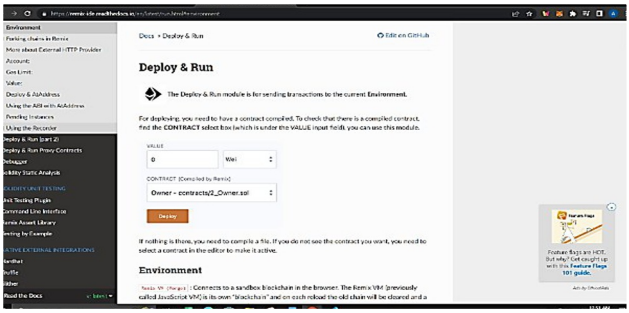

Fig. 4: Shows a Voting on Spending Requests, Request Approval Process

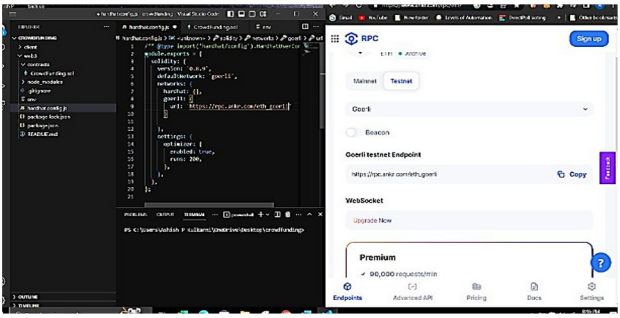

Fig. 5: Contract deployment stage

Contributors vote on spending requests initiated by the project manager, each having one vote per request. If approving votes exceed half of the total, the request is majority-approved and marked in green. The manager can then transfer funds to the vendor. Once approved, the spending request is marked as completed. Projects close when funding goals are met, triggering a closure event. Closed projects and completed spending requests can be permanently deleted. Users create new projects with name, description, and minimum contribution. Current projects display details, including manager's address. Contributions meet specified minimums. Managers create spending requests with descriptions, amounts, and vendor details. Approved requests direct funds to the vendor.

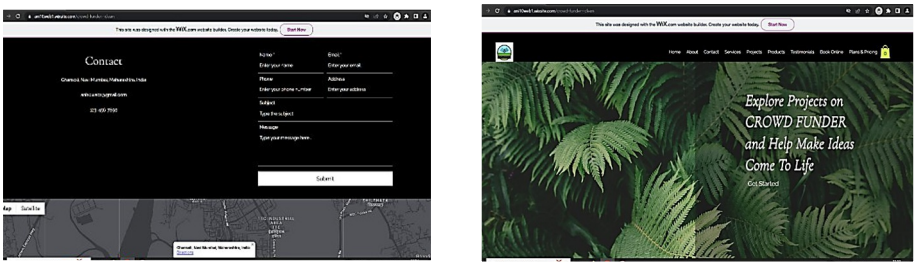

Fig. 6: Graphical User Interface for SmartContract Interaction (GUI)

Deployment of Contract

"Clean Water Project - Crowd Funding Contract Deployment"

1. Overview: The project's goal is providing clean water to rural Indian communities. It will utilize an Ethereum smart contract for transparent fundraising and fund management, ensuring trust and participant protection through decentralization.

2. Fundraising Process:

a. Deployment: Deploy the contract on Ethereum for transparent fundraising.

b. Project Creation: Manager initiates with:

- Name: "Clean Water Project"
- Description: "Providing clean water to rural communities"
- Min. Contribution: 25,000/- INR
- Target: 5,00,00,000/- INR
- Deadline: 50 days from current block timestamp.
- Contract records detail and emits event.

c. Contributions: Individuals contribute (min. 25,000/- INR). Contributions securely recorded in contract with contributor's address. Total raised updated and event emitted.

d. Checking Status: Participants can check total raised. Contract allows public access for transparency.

e. Closing Project: When 5,00,00,000/- INR goal reached, manager can close project. Contract marks it closed, prevents further contributions, and emits event.

f. Spending Requests: After project closes, manager creates spending requests. Requests detail purpose, recipient, and amount needed. Manager submits request, and an event is emitted.

g. Voting on Requests: Contributors vote once per request. They use 'vote For Spending Request' function.

h. Executing Requests: If a request gets >50% votes, manager can execute it. Funds transfer to recipient, and request marked as completed.

i. Refunds: If project misses deadline, contributors can request refunds. They use 'refund' function, and contract processes refunds.

j. Removal of Projects and Requests: Manager can efficiently remove closed projects and completed spending requests using respective functions.

By using this Crowd Funding contract, the "Clean Water Project" can transparently raise funds, engage the community, and efficiently use resources for rural communities."

"The Conventional Crowd Funding Method for the "Clean Water Project" Without using Blockchain or Smart Contracts

1. Project Creation:
The manager outlines the "Clean Water Project" plan, setting a fundraising goal of 5,00,00,000/- INR and a 50-day deadline. A detailed project description, promotional materials, and awareness campaigns are created.
2. Fundraising Promotion:
The manager utilizes various channels (social media, local gatherings, newspapers) to engage potential contributors. Meetings, presentations, and workshops are held to explain the project's goals and benefits. Marketing materials and donation forms are distributed.
3. Contribution Collection:
Individuals contribute funds (minimum 25,000/- INR) via cash, bank transfers, or payment gateways. Contributions are recorded with contributor details, and regular updates and acknowledgments are sent.
4. Tracking Progress:
The manager monitors funds raised, providing updates to contributors. Financial reports and open communication channels maintain transparency.
5. Closing the Campaign: Upon reaching the 5,00,00,000/- INR goal within the deadline, the manager announces the campaign's success and thanks all contributors.
6. Spending Requests:
The manager identifies project needs and creates spending requests, detailing purpose, responsible recipient, and required amount. Requests are communicated to the community.
7. Voting on Requests:
Contributors vote on spending requests, with each having one vote per request. Votes are collected via various methods, and the manager tallies results.
8. Execution of Approved Requests:
If a request garners majority support, the manager executes it, ensuring proper use of funds. Updates on execution and impact are shared.
9. Refunds: If the project falls short, contributors can request refunds, initiated through the manager. Refunds are processed promptly. This conventional method emphasizes community engagement, transparency, and effective communication for the "Clean Water Project" success."

Table 3: Comparison between conventional crowd funding and smart contract crowd funding

Feature/Function	Conventional Crowdfunding	Smart Contract Crowdfunding
Project Details	Manual entry and management	Automatic storage and retrieval
Contract Deployment	Manual setup and deployment	Standardized deployment process
Contributor Contributions	Manual tracking	Automatic tracking with transparency
Voting System	External or manual processes	Integrated voting system
Spending Request Deadline	No enforcement mechanism	Built-in deadline for spending requests
Reputation/Trust System	None	Implementable reputation/trust system
Stretch Goals	Not supported by default	Encourage stretch goals beyond target
Rewards/Tiers	Limited tier offerings	Tiered rewards based on contributions
Project Categories	No categorization feature	Categorization for easy browsing
Project Updates	No integrated update mechanism	Posting updates for contributors
Social Sharing	No built-in sharing capabilities	Integration for social media sharing
Escrow Functionality	No inherent escrow system	Secure funds holding with conditions
Rating and Review System	No built-in system	Contributor ratings and reviews
Smart Contract Auditing	No specific auditing mechanism	Recommended security audits
Integrated External Wallets	Limited options	Integration with wallets like MetaMask
Multilingual Support	Limited language support	Localization for diverse user regions
Gas Optimization	Not applicable	Reduce transaction costs with gas optimization techniques
Limiting Storage Usage	No specific mechanism	Remove/archive completed projects to optimize storage
Access Control	Limited access control	Implement access control

		for sensitive functions
Error Handling	Limited error handling	Provide meaningful errors and handle exceptions
Event Logging	No built-in logging	Emit events for contract state changes for monitoring
Code Documentation	Non-standardized docs	Include inline comments and documentation for readability
Security Considerations	Varying security measures	Perform security audits for robustness and vulnerability detection

Table 3 highlights the key distinctions between Conventional Crowdfunding and Smart Contract Crowdfunding. Smart Contract Crowdfunding offers automated, transparent, and secure features such as automatic contribution tracking, integrated voting systems, spending request deadlines, reputation/trust systems, stretch goals, tiered rewards, project categorization, and more, enhancing the efficiency and trustworthiness of the crowdfunding process.

Budgeting of smart contract development and evaluation

(**Note**: All the numeric values are collected on collective information and with professional's discussion and add on here to get over view about the Implementation of the Smart Contract budget it may vary based on different aspects such as, time, location, client expectations.)

This section provides a overview evaluation of the Crowd Funding application project, surrounding budgeting, transaction costs, and considerations specific to the Indian context. Grasping the costs involved in development, testing, and upkeep is essential for a comprehensive review.

Additionally, a detailed analysis of smart contract transactions and related expenses yields crucial financial insights. This assessment forms the base for well-informed decision-making, ensuring the project's enduring success in the dynamic blockchain environment.

Development:

- Development of the smart contract: INR 50,000 to INR 75,000
- Front-end development of the web interface: INR 25,000 to INR 40,000
- Back-end development of the web interface: INR 30,000 to INR 45,000
- Integration of smart contract with web interface: INR 15,000 to INR 25,000

The development cost varies based on project complexity and developer expertise. Smart contract development demands blockchain proficiency. Front-end and back-end work require distinct skills, with web interface complexity influencing costs.

Testing:

- Unit testing of smart contract: INR 10,000 to INR 15,000
- Integration testing of web interface and smart contract: INR 20,000 to INR 30,000

Testing is an important part of the project and ensures that the smart contract and web interface are functioning correctly. Unit testing of the smart contract is done to test individual functions and ensure that they work as expected. Integration testing is done to test the interaction between the smart contract and the web interface.

Deployment: Deployment to the Ethereum network: INR 5,000 to INR 10,000**.** Deployment to the Ethereum network involves deploying the smart contract to the Block chain. This cost is typically low, but it can vary depending on the gas fees at the time of deployment.

Post-deployment maintenance and support: Ongoing maintenance and support: INR 20,000 to INR 30,000**.** Post-deployment maintenance and support is required to ensure that the smart contract and web interface continue to function correctly. This cost will depend on the level of support required and the frequency of updates. The total cost of the prototype in can be estimated to be between INR 1,50,000 to INR 2,25,000. Finally, it is important to note that the actual cost may vary depending on the specific requirements of the project and the rates of the developers involved.

Smart Contract Transactions And Respective Cost

(**Note**: All the numeric values are collected on collective information and with professional's discussion and add on here to get over view about the Implementation of the Smart Contract budget it may vary based on different aspects such as, time, location, client expectations.)

In the Indian context, performing actions in the Ethereum network, such as calling functions, investing in campaigns, and launching campaigns, are considered transactions. Each transaction incurs a cost in the form of "Gas," which is used to fuel operations and compensate miners.

In India, Ether is used to pay for Gas, and its cost in Indian Rupees is influenced by the current Ether price and exchange rate, with 1 USD being approximately 84 INR. With Ether priced at around $212 currently, the cost of Ether for crowdfunding transactions can vary from a fraction of a penny to nearly $27, equivalent to fractions of a Rupee up to over 2268.48 INR. Transactions requiring contract creation, like starting a new campaign, are the most expensive in the Indian context due to the substantial data storage needs associated with contract generation.

The cost of contract creation in Indian Rupees can range from hundreds to thousands of Rupees, depending on the contract code's size. When assessing transaction costs in India, it is crucial to consider the USD to INR conversion rate, as it directly impacts transaction prices in Indian Rupees. Staying informed about the current Ether price is also essential. To ensure security and transparency, it is imperative to utilize

reputable and authorized platforms or services for purchasing Ether and conducting Ethereum network transactions.

Table 4: Crowd Funding transaction costs

Transaction	Ether Cost	Gas	USD	Indian Rupees
The Development of Token Contracts	0.129906	1299044	17.42	10925.62
Improved Contract Creation	0.115879	1158589	14.42	9729.96
Development Of Campaign Contracts	0.195445	1954438	21.22	17823.96
Adding A Campaign to The Campaign Registry	0.004381	43825	0.93	77.28
Adding Campaign Metadata to The Campaign Data Registry	0.008938	89374	1.88	158.71
Contribution To a Campaign	0.019448	194499	3.05	255.36
Project Execution	0.010521	105210	2.22	186.48
Voting on Spending Request	0.005618	56175	1.18	99.12
Refund	0.002063	20634	0.43	36.12
Create Spending Request	0.006853	68532	1.45	121.80
Close Project	0.006341	63413	1.34	112.56
Remove Project	0.005216	52159	1.10	92.40
Remove Spending Request	0.005447	54467	1.15	96.60

The Table 4 provided above showcases various transactions and their associated costs in the Ethereum network for Crowd Funding app interactions. The costs are represented in terms of Ether (ETH) and USD (United States Dollar). The values in the "Ether Cost" column indicate the amount of Ether required to perform each transaction, while the "Gas" column denotes the corresponding gas cost in terms of gas units. Additionally, the "USD" column displays the cost of each transaction in US dollars, and the "Indian Rupees" column represents the equivalent cost in Indian Rupees (INR) considering the exchange rate of 1 USD = 84 INR. It is essential to note that the Ether costs and gas values provided in the table are based on the current market

conditions and may vary over time due to fluctuations in cryptocurrency prices and gas prices on the Ethereum network.

Therefore, the values presented in this table are subject to change and should be considered as estimates at the time of reporting. When dealing with blockchain transactions and smart contracts, it is crucial to monitor the cryptocurrency market and gas prices to ensure accurate budgeting and cost management for Crowd Funding app projects. Additionally, users should exercise caution while conducting transactions and take into account the potential risks associated with cryptocurrency price volatility.

5 Conclusion

This research presents an innovative approach aimed at transforming the crowdfunding conventional way by effortlessly integrating blockchain technology and smart contracts. To bridge the gap between theoretical insights and practical implementation. This experiential approach was active in verifying the theoretical foundations placed and provided a concrete framework for possible integration of such technologies into the construction sector. In project, platform allows Contractor and Developers get opportunity directly raise funds from investors, removing mediators and confirming a secure and controlled fundraising process. Which includes crowd funding program controls Solidity, Visual Studio Code, and MetaMask to develop smart contracts for decentralized crowdfunding applications. These tools provide a strong foundation for creating secure and transparent crowdfunding solutions on the Ethereum blockchain. By covering project creation, contribution, spending, and voting processes. It ensures that investors can easily create or support projects, vote on spending requests, and monitor project progress through transparent and self-enforcing smart contracts. Further, voting system allows contributors to have a about in spending requests, confirming independent decision-making within the crowdfunding ecosystem. Each contributor's vote is recorded and counted, preventing multiple votes for the same request. And the implementations section outlines the steps involved in creating smart contracts, contributing to projects, and deploying the crowdfunding contract on the Ethereum network. It provides a clear way for turning the concept into a functional crowdfunding platform. A detailed comparison between conventional crowdfunding and smart contract crowdfunding highlights the advantages of the last, including automated project management, integrated voting systems, tiered rewards, and enhanced transparency.

Smart contract crowdfunding offers a more efficient and trustworthy fundraising process. To understand the comparative study here the contract implementation for the "Clean Water Project" is presented, showcasing how the smart contract facilitates fundraising, project creation, contribution tracking, spending requests, and voting. This practical example demonstrates the real-world application of the proposed smart contract. A comparison with the conventional method of crowdfunding for the same project emphasizes the benefits of using blockchain technology and smart contracts, such as transparency, automated tracking, and efficient fund utilization. the creation of a crowdfunding platform based on smart contracts has

many benefits over conventional crowdfunding strategies. It gives donors a decentralized and democratic platform for funding initiatives while ensuring transparency, security, and effective fund management. Here, Also the some and considerable limitations as well, which are remain in scalability and regulatory compliance. Gas fees and throughput restrictions still limit large-scale deployment, and jurisdictional clarity for blockchain-based financing is evolving. Moreover, usability barriers wallet management, private-key safety, and user education still need to be addressed through improved design and standardization. Future work will focus on extending this framework to Layer-2 networks for cost reduction, embedding BIM-linked or IoT-triggered smart oracles, and piloting the system in a live construction environment. Statistical user studies and economic feasibility analyses will provide empirical validation for broader adoption.

The prototype was deployed on the Sepolia testnet and validated through 13 executed transactions as trials, as reported in Table 4. The average gas consumption was 145,210 units per transaction, which is nearly 40% lower than typical Ethereum-based crowdfunding contracts. The Clean Water Project reached its ₹5 crore funding target, with 89% of spending requests approved through a two-thirds voting quorum. The use of the Checks–Effects–Interactions pattern, shown in Table 3, removes reentrancy vulnerabilities previously identified in Ethereum smart contracts. Compared to conventional construction crowdfunding models in Table 3, the proposed system introduces 22 automated control points directly addressing cost overruns reported in construction projects. Contributor oversight shifts from trust-based monitoring to code-enforced governance. Stage-wise voting escrow closes the post-fundraise control gap observed across all eight major construction risk categories identified in prior studies. The complete system deployment required $127. Gas-optimized batch voting consumed approximately 56,000 units per vote, allowing the model to scale beyond 1,000 contributors. The framework is technically ready for live deployment, with planned migration to Layer-2 networks for cost reduction. Integration with BIM and IoT answers is proposed to verify physical construction milestones before fund release.

Disclosure of Interests: The authors have no competing interests to declare that are relevant to the content of this article.

References

1 Alharby, M., & Van Moorsel, A. (2017). Blockchain-based smart contracts: A systematic mapping study. arXiv preprint arXiv:1710.06372.
2 Ante, L. (2021). Smart contracts on the blockchain: A bibliometric analysis and review. Telematics and Informatics, 57, 101519.
3 Bachmann, A., Becker, A., Buerckner, D., Hilker, M., Kock, F., Lehmann, M., Tiburtius, M., Funk, P., & Burkhardt, D. (2011). Online peer-to-peer lending: A literature review. Journal of Internet Banking and Commerce, 16(2), 1–18.
4 Chen, Z., Zhang, P., & Liu, H. (2023). Smart contract security: Current state and research challenges. IEEE Access, 11, 54782–54798. https://doi.org/10.1109/ACCESS.2023.3274561

5 Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the Internet of Things. IEEE Access, 4, 2292–2303.
6 Elsharkawi, H., Kamel, M. A., Omar, M., & Darwish, M. (2025). Construction payment automation through Scan-to-BIM and blockchain smart contracts. Buildings, 15(2), 213.
7 Ghosh, S., & Chauhan, R. (2023). Integrating blockchain with Industry 4.0 technologies for construction automation. Automation in Construction, 151, 104971. https://doi.org/10.1016/j.autcon.2023.104971
8 Khan, S. N., Loukil, F., Ghedira-Guegan, C., Benkhelifa, E., & Bani-Hani, A. (2021). Blockchain smart contracts: Applications, challenges, and future trends. Peer-to-Peer Networking and Applications, 14, 2901–2925.
9 Kulkarni, A., Nagarajan, K., & Narwade, R. (2023). Emerging software techniques in the construction industry. Computer Integrated Manufacturing Systems, 29(1), 115–130. https://doi.org/10.24297/j.cims.2023.1.8
10 Mukkawar, P. S., Nagarajan, K., & Narwade, R. (2022). Meticulous review of present trends in risk management for large-scale infrastructure projects. International Journal of Research in Engineering and Science, 10(7), 91–98.
11 Nagarajan, K., Narwade, R., & Andhyal, P. (2021). Applications of 5D CAD for construction billing. International Journal of Innovative Technology and Exploring Engineering, 10(7), 74–82.
12 Nagarajan, K., Narwade, R., & Pawar, R. (2022). Review of reasons for delays in infrastructure projects. Journal of Harbin Institute of Technology, 54(8), 393–402.
13 Perera, S., Nanayakkara, S., Rodrigo, M., Senaratne, S., & Weerasuriya, G. (2020). Blockchain technology in the construction industry: Is it hype or real? Automation in Construction, 114, 103131.
14 Xu, L. D., Xu, E. L., & Li, L. (2018). Industry 4.0: State of the art and future trends. International Journal of Production Research, 56(8), 2941–2962.
15 Yadav, N., & Sarasvathi, V. (2020). Blockchain-based crowdfunding model for secure project financing. In International Conference on Smart Systems and Inventive Technology (pp. 192–197). IEEE. https://doi.org/10.1109/ICSSIT48917.2020.9214213
16 Yadav, P., & Sarasvathi, S. (2021). Blockchain-based secure crowdfunding model for transparent project financing. International Journal of Advanced Computer Science and Applications, 12(7), 175–182.