

Scientiarum: A Multidisciplinary Journal
Vol. 2 No. S1 (2026): Supplement Issue
DOI: 10.54646/SAPARS.2026.85

SAPAR SERVICES LLP
Scholarly Articles Publication And Repository Services

CROWDFUNDING USING BLOCKCHAIN

Mr. Manjunatha S[1] (Project Guide, Head of the Department) | Mr. Guna Sagar R[2]

sagarguna158@gmail.com

Department of Master of Computer Applications (MCA), Academic Year 2025–26

S.E.A. College of Science, Commerce and Arts, K R Puram, Bengaluru – 560049, Karnataka, India

ABSTRACT

Crowdfunding lets many people support an idea, a cause or a young enterprise with small contributions. Most existing platforms, however, rely on a central intermediary that charges service fees, controls the movement of money and asks contributors to trust both the platform and the campaign creator with limited visibility into how the funds are used. This paper presents *Crowdfunding Using Blockchain*, a decentralized web application in which the rules and the funds of each campaign are governed by an Ethereum smart contract instead of a platform operator. A campaign creator sets a funding goal, a deadline and a description; contributors pledge Ether directly from a MetaMask wallet; and every contribution is recorded permanently on the blockchain. By design, funds are released only under conditions agreed in advance: withdrawal requests require the approval of a majority of contributors, and contributors may reclaim their money if the goal is not met by the deadline. The interface is built with React.js and Tailwind CSS, the contract is written in Solidity, Hardhat is used for compiling, testing and deploying, and Ethers.js connects the interface to the network.

The paper describes the problem, the existing systems, the proposed architecture, the design models, the implementation and the testing carried out. The implemented results consist of working application screens captured on an Ethereum test network and fourteen functional test cases, all recorded as passed. No large-scale performance measurements or formal security audit are reported, so benefits such as lower fees and greater transparency are presented as expected

outcomes of the design rather than as measured results.

KEYWORDS: *blockchain; crowdfunding; Ethereum; smart contracts; Solidity; decentralized application; MetaMask; transparency.*

INTRODUCTION

Crowdfunding has become a familiar way of raising money for social causes, creative work and early-stage ventures. Well-known platforms such as Kickstarter, Indiegogo and GoFundMe make it easy to reach a large audience, but they do so by acting as intermediaries. They host the campaigns, process the payments and decide the terms on which money changes hands. For contributors this means trusting the platform and, in most cases, the campaign creator as well. For creators it means paying fees and accepting the availability and regional limits of a service they do not control.

Blockchain technology offers a different way to organize this trust. A blockchain is maintained by a distributed network of nodes rather than by a single authority. When a contribution is made, the transaction is broadcast to the network, validated by the nodes using cryptographic verification, and grouped into a block that carries its own hash together with the hash of the previous block. Once the block is added to the chain it forms a permanent and effectively immutable record. Three properties follow from this arrangement and matter directly for crowdfunding: decentralization, because no single party controls the funds and the rules are enforced by code; transparency, because every participant can verify contributions and fund usage; and immutability, because recorded data cannot be quietly altered. Researchers have therefore examined the use of blockchain and smart contracts to improve the security, transparency and efficiency of crowdfunding [1, 2, 3].

This paper presents a decentralized crowdfunding platform built on the Ethereum blockchain. Each campaign is managed by a smart contract that acts as an autonomous governing body. The creator defines the funding goal, deadline and description, contributors pledge Ether directly to the contract, and the contract tracks contributions and validates deadlines. In the intended design, the creator cannot simply take the money: a withdrawal request is put to the contributors, and funds are

released only if a majority approves. If the goal is not reached by the deadline, contributors may withdraw their funds so that nothing is lost. The same approach can serve social causes, start-ups, creative initiatives and community-driven projects.

The remainder of the paper is organized as follows. Sections 2 and 3 state the problem, the objectives and the scope. Sections 4 to 6 review related work, the existing practice and the proposed system. Sections 7 and 8 describe the architecture, methodology, design models and implementation. Section 9 reports the testing and results, keeping what has been implemented and observed clearly separate from what is merely expected. Sections 10 to 12 discuss advantages, limitations, future work and conclusions.

PROBLEM STATEMENT

Centralized crowdfunding platforms share a set of weaknesses that follow from their structure. Service fees are charged for hosting and processing transactions, which reduces the amount that reaches the campaign. Contributors have little insight into how funds are allocated and used, so they must rely on the platform and on the creator, and the risk of fraud or misuse remains with them. Geographical restrictions can exclude people from taking part, and because campaigns and funds depend on the platform, any shutdown or technical failure on its side affects everyone involved.

The problem addressed here is therefore to design a crowdfunding platform in which (i) the rules for collecting and releasing money are enforced without a trusted intermediary, (ii) contributions and withdrawals can be verified publicly, (iii) a campaign creator cannot withdraw funds without the consent of the people who provided them, (iv) contributors are protected if a campaign fails to reach its goal, and (v) anyone with an internet connection and a cryptocurrency wallet can take part. The solution must also remain usable: an interface that hides unnecessary blockchain detail is as important as the contract that sits behind it.

OBJECTIVES AND SCOPE

The overall aim is to build a decentralized, secure and transparent fundraising platform. Seven specific objectives follow from the problem: to remove the need for central authorities in

fundraising; to record all contributions and withdrawals on the blockchain for public verification; to prevent fraud and unauthorized use of funds through smart contracts; to let contributors approve or reject withdrawals by voting; to reduce service charges compared with centralized platforms; to enable worldwide participation through cryptocurrency; and to ensure that creators can use funds only when the backers approve.

The scope covers five areas: campaign creation and management with a goal, deadline and description; blockchain-based tracking of every contribution; smart-contract automation of rules such as refunding contributors when a goal is not met; voting-based approval of withdrawal requests; and a design that allows later extension to multiple currencies, mobile applications and cross-chain operation. Legal and regulatory questions around crowdfunding and cryptocurrency lie outside the scope of this paper.

LITERATURE REVIEW

Several studies have explored the combination of blockchain and crowdfunding. Saadat et al. [1] proposed a crowdfunding system based on Ethereum smart contracts in which funds are released automatically once predefined conditions are met. Garg et al. [2] reviewed how blockchain can address fraud and security concerns in crowdfunding transactions, and Gupta et al. [3] developed a decentralized application that uses the security features of blockchain to strengthen donor trust. Chattani et al. [4] also discuss crowdfunding with blockchain technology. The same body of work notes that blockchain-based crowdfunding faces regulatory concerns, scalability issues, a need for user education and the computational cost of blockchain transactions [2, 3].

More recent platform-oriented studies are compared in Table I. They share a common pattern: Ethereum-style smart contracts remove intermediaries and improve transparency, while the limitations reported are mostly practical, namely scalability, regulation, cost and technical complexity for ordinary users.

Ref.	Approach	Reported outcome	Limitation noted
[1]	Decentralized platform for social and educational causes using	Secure, transparent donations; donors can track	Limited scalability; regulatory concerns

Ref.	Approach	Reported outcome	Limitation noted
	Ethereum and Polygon smart contracts	fund utilization	
[2]	Ethereum blockchain with smart contracts for security and decentralization	Improved transparency and security of transactions	Smart-contract operations may seem complex to users
[3]	Ethereum-based decentralized application with Solidity smart contracts	Enhanced security, no middlemen, and transparency	High initial costs; technical challenges
[4]	"Easy-Starter," a peer-to-peer crowdfunding website with Solidity contracts, automated transfers, and community approvals	Fewer intermediaries, lower transaction fees, improved security and transparency	Adoption affected by technical complexity, gas fees, and reliance on wallets such as MetaMask
[5]	Design-science study with interviews and a prototype of blockchain tokens for equity crowdfunding	Equity-token prototype improving efficiency, transparency, and secondary-market trading	Limited to Ethereum; needs further validation for private equity or venture capital

Taken together, these works support the central idea of this paper, that smart contracts can replace much of the trust normally placed in a platform. The present work focuses on a complete but compact workflow in a web application: creating a campaign, contributing through a wallet, and linking the release of funds to the consent of contributors.

EXISTING SYSTEM

In the existing model, platforms such as Kickstarter, Indiegogo and GoFundMe operate centrally and stand between campaign creators and contributors. They provide easy access to funding, yet several drawbacks recur. High service fees are charged for hosting and processing transactions. Fund utilization and allocation are not transparent to contributors. Fraud or misuse is possible, since contributors must fully trust both the platform and the creator. Participation can be limited by geographical restrictions. Finally, campaigns and funds depend on the availability of the platform

itself, so if it shuts down or runs into difficulty, the campaigns and money held there are affected.

PROPOSED SYSTEM

The proposed system removes the dependency on a centralized platform by using a decentralized, blockchain-based architecture. Campaign data and transactions are stored on the Ethereum blockchain, which provides transparency, security and immutability. Creators and contributors interact directly, and smart contracts automate fund management so that money is released only when predefined conditions, such as contributor approval, are satisfied. Anyone who has an internet connection and a crypto wallet can participate, and the removal of the intermediary is expected to lower transaction and service charges.

The process works as follows. A creator launches a campaign by entering the details and confirming the transaction in MetaMask. Contributors browse campaigns and pledge Ether, which is held by the smart contract and recorded on the chain. When the creator wants to use the money, a withdrawal request is raised; contributors vote on it, and the contract releases the funds only if a majority approves. If the funding goal is not met by the deadline, contributors may withdraw their funds. Table II summarizes how this differs from the centralized model. The comparison is deliberately balanced: the proposed system gains in decentralization, transparency and security, but it asks users to be familiar with cryptocurrency, whereas existing platforms are friendlier to users who pay in conventional currency.

TABLE II: EXISTING SYSTEMS VERSUS PROPOSED SYSTEM

Feature	**Proposed system**	**Existing systems**
Decentralization	Fully decentralized (blockchain)	Centralized
Transparency	Blockchain-based, real time	Relies on fundraisers
Global accessibility	High (cryptocurrency support)	Moderate (supports fiat currency)
Approval mechanism	Approvers through smart contracts	Managed internally
Security	Blockchain-native	Traditional web security
Ease of use	Requires cryptocurrency familiarity	User-friendly for fiat users

ARCHITECTURE AND METHODOLOGY

System Architecture

Fig. 1 shows how the parts of the system fit together. The user, acting as a campaign creator or a contributor, works through an ordinary web browser to create campaigns, contribute funds and approve withdrawals. MetaMask, a browser-based wallet, provides authentication, wallet management and transaction approval, so users send and receive funds directly from their own accounts. The user interface is developed with React.js and styled with Tailwind CSS; it displays campaigns, handles user input and shows transaction updates.

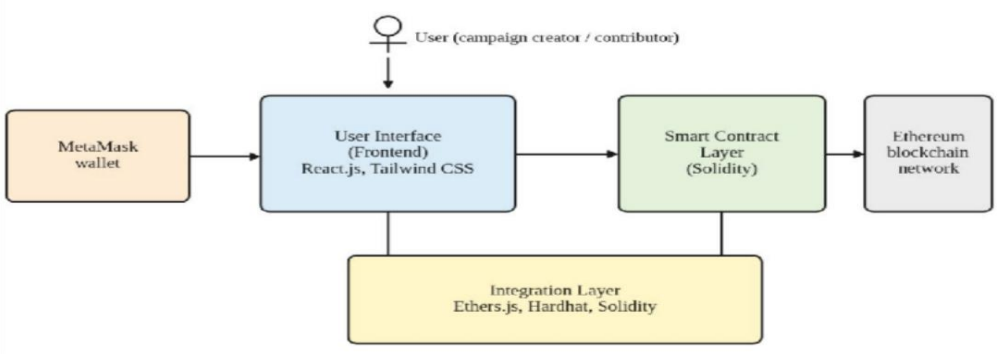


Fig. 1. System architecture of the proposed crowdfunding platform.

Between the interface and the chain sits the integration layer. Solidity is used to write the smart

contracts, Ethers.js establishes communication with the blockchain, and Hardhat compiles, tests and deploys the contracts. The smart-contract layer holds the business logic, namely campaign creation, fund collection, contributor voting and the conditional release of funds, and executes it without third-party intervention. The Ethereum blockchain network, an Ethereum test network during development, stores every transaction permanently, which gives the process its transparency, immutability and trust.

Workflow

The overall working process is shown in Fig. 2. A campaign is created first, and contributions are then added to it. When a withdrawal is requested, the decision branch leads either to the release of funds or, if no request is made, to the end of the process. In practice the same path is followed in every case: the user opens the application in a browser, connects MetaMask, and starts an action such as creating a campaign or contributing. The interface passes the action to the smart contracts through Ethers.js. The contracts, which are compiled, tested and deployed with Hardhat on an Ethereum test network through Infura, carry out campaign management, fund collection and withdrawal handling, and each transaction is validated and permanently recorded on the blockchain. The release of funds in the flowchart is subject to the contributor approval described in Section 6.

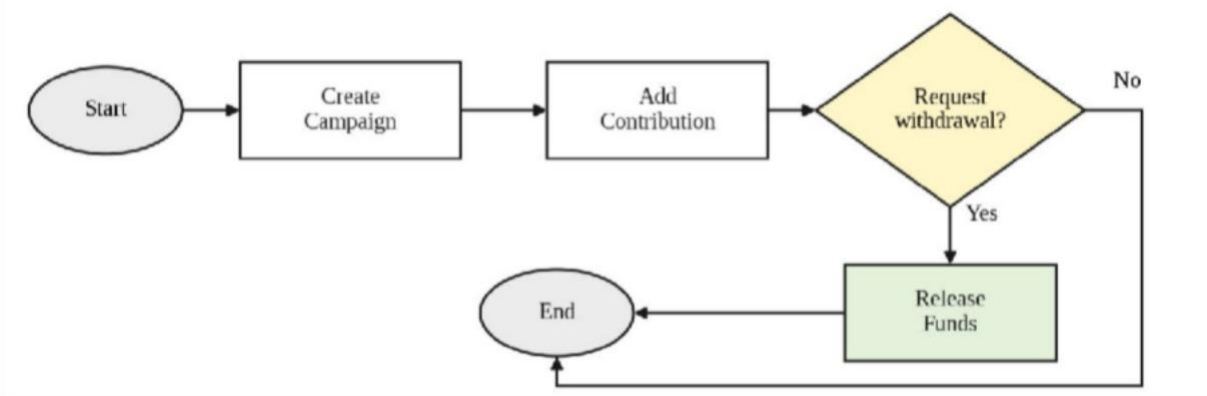


Fig. 2. Workflow of the crowdfunding process from campaign creation to release of funds.

Methodology and Requirements

The work began with a feasibility study covering four aspects. Technically, the system relies on widely used, open-source technologies (Ethereum, Solidity, React.js, Hardhat, Ethers.js and MetaMask) with a large developer community and ordinary hardware. Operationally, a user-friendly interface and the trust created by the blockchain are expected to encourage adoption, and once deployed most processes are handled by smart contracts. Economically, the use of open-source tools keeps development cost low, and gas fees are paid only when transactions occur. In terms of time, the modular design allows the interface, the contracts and the integration to be developed in parallel under an Agile approach.

The functional requirements are campaign management (creating campaigns and showing details such as contributors and funds raised), contributions in cryptocurrency with visible transaction records, withdrawal requests that need approval from contributors, wallet connectivity through MetaMask, and worldwide accessibility. The non-functional requirements are security through tamper-proof records and contracts that prevent unauthorized manipulation of funds, transparency through publicly verifiable transactions, scalability for many simultaneous users, and usability through an intuitive interface. The development environment is a standard computer with an Intel Core i5 or equivalent processor, at least 8 GB of RAM, a 256 GB SSD and a high-speed internet connection, running Windows 10 or 11.

DESIGN AND IMPLEMENTATION

Use Case Model

Fig. 3 consolidates the use case diagrams of the design. Three kinds of user take part, together with the server. The public viewer can view campaigns and transactions and give encouragement or support points. The fund raiser starts a new campaign by filling in a form, updates the progress of the cause, raises a withdrawal request by entering the details and the required amount, and can end a campaign by giving a reason. The backer selects a campaign, enters a funding amount and contributes, with the contribution stored in the smart contract. In every action performed by the fund raiser or the backer, authorization takes place through the MetaMask wallet. The server shows campaigns and transactions, stores the support points, and registers campaigns and

withdrawal requests.

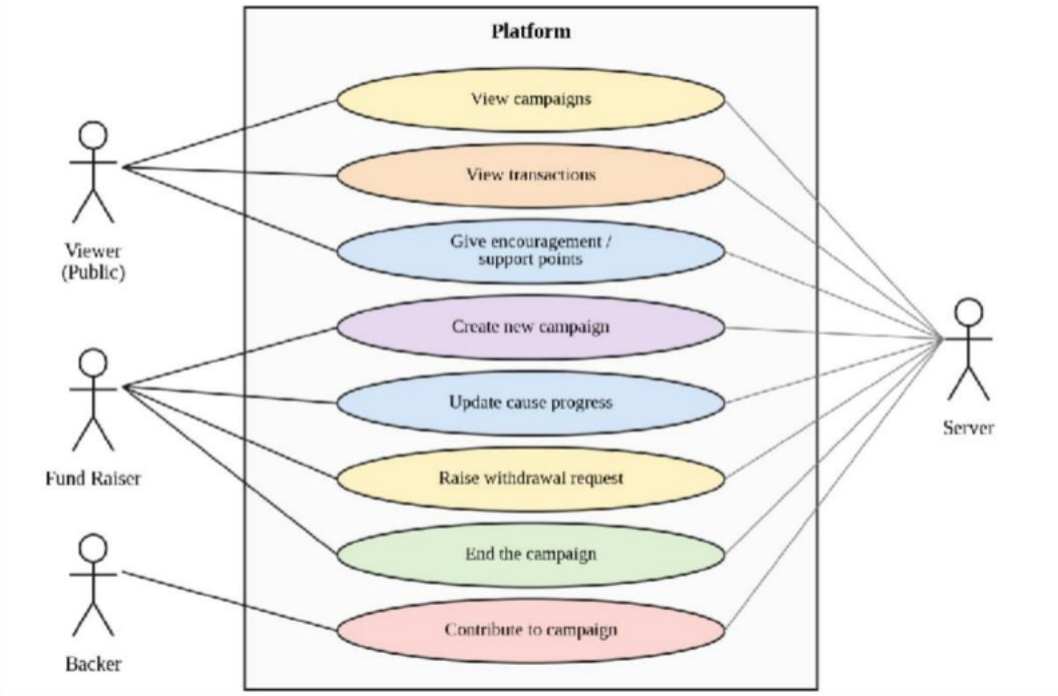


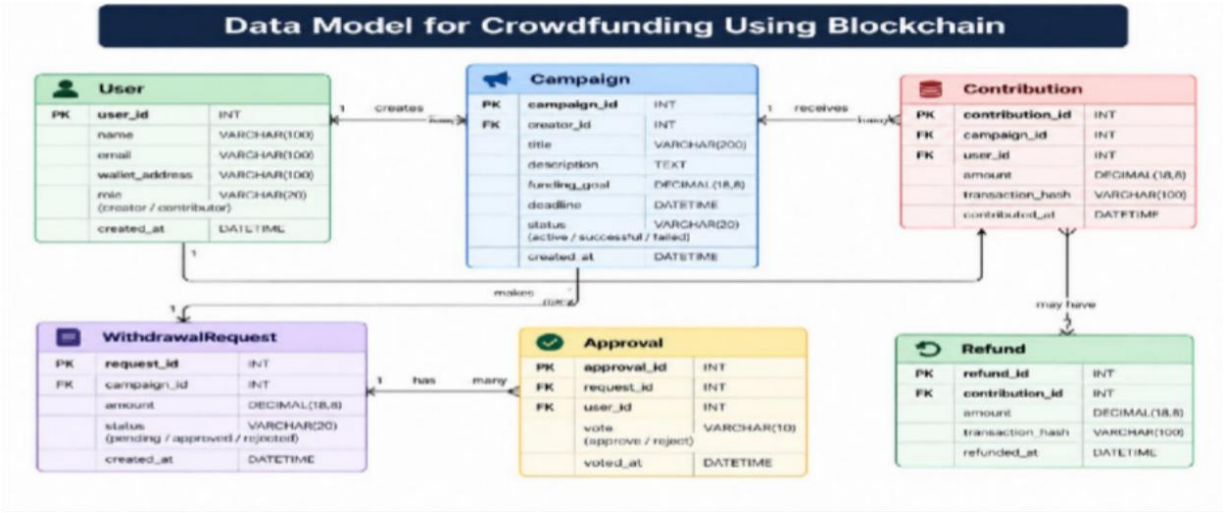
Fig. 3. Use case diagram of the platform. Fund raiser and backer actions are authorized through the MetaMask wallet.

The design documentation adds that each contributor has the right to approve or deny a withdrawal request. When the request specifies an amount, a reason and a recipient wallet address, it is forwarded to the contributors, and the smart contract executes the transfer only when a majority approves. The smart contract itself behaves as an automated actor that secures funds, executes transactions and keeps the record transparent.

Data Model

Blockchain storage is not a relational database, but the data handled by the contract can still be described in entity–relationship terms. Fig. 4 presents this logical model, derived from the structure stored by the smart contract and from the class diagram of the design. A WALLET ACCOUNT is identified by its address, which also serves as the owner of a campaign, the donor of a contribution and the approver of a request. A CAMPAIGN stores its owner, title, description, funding target, deadline, collected amount, image and an active flag. Each DONATION records the donor and the amount for one campaign; in the contract this is held in two parallel lists of

donors and donations. A REQUEST represents a withdrawal request with its description, value, recipient, completion state and approval count. One wallet may create many campaigns and make many donations, one campaign receives many donations and may have many requests, and a request can be approved by many wallets.



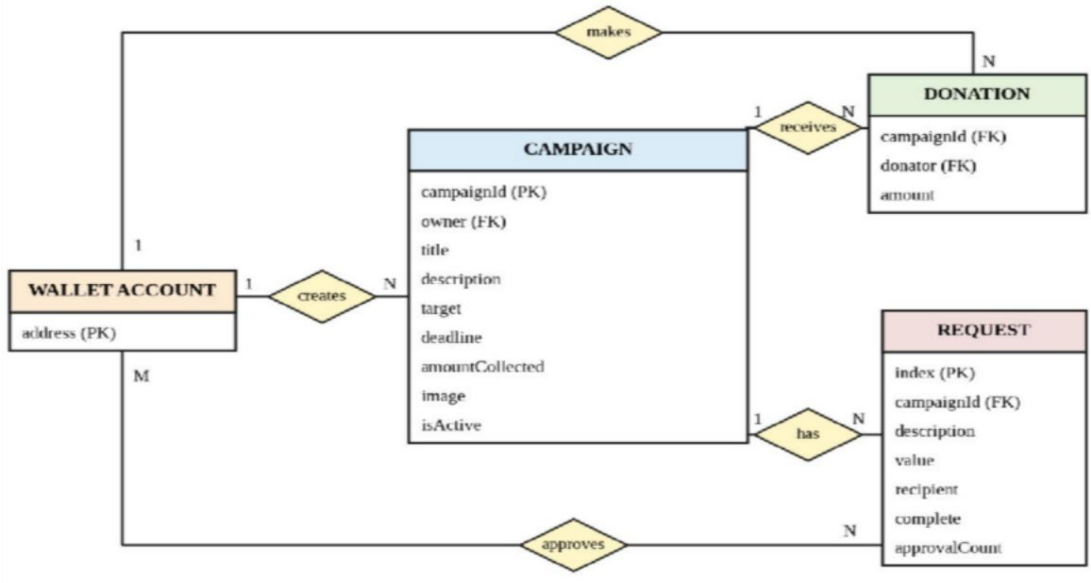


Fig. 4. Logical entity–relationship model of the data handled by the smart contract. PK and FK denote primary and foreign keys.

Contribution Transaction Flow

The sequence of a contribution is shown in Fig. 5. After the user selects a campaign to fund, the platform asks MetaMask to connect the wallet and the user approves the request. The user then enters an amount that must reach the minimum contribution, the platform sends an authorization

request, and the user approves it in the wallet. MetaMask submits the Ether transaction to the smart contract, which records the contribution; the platform then refreshes and shows the updated status. If the wallet balance is insufficient, the case is handled by the wallet itself.

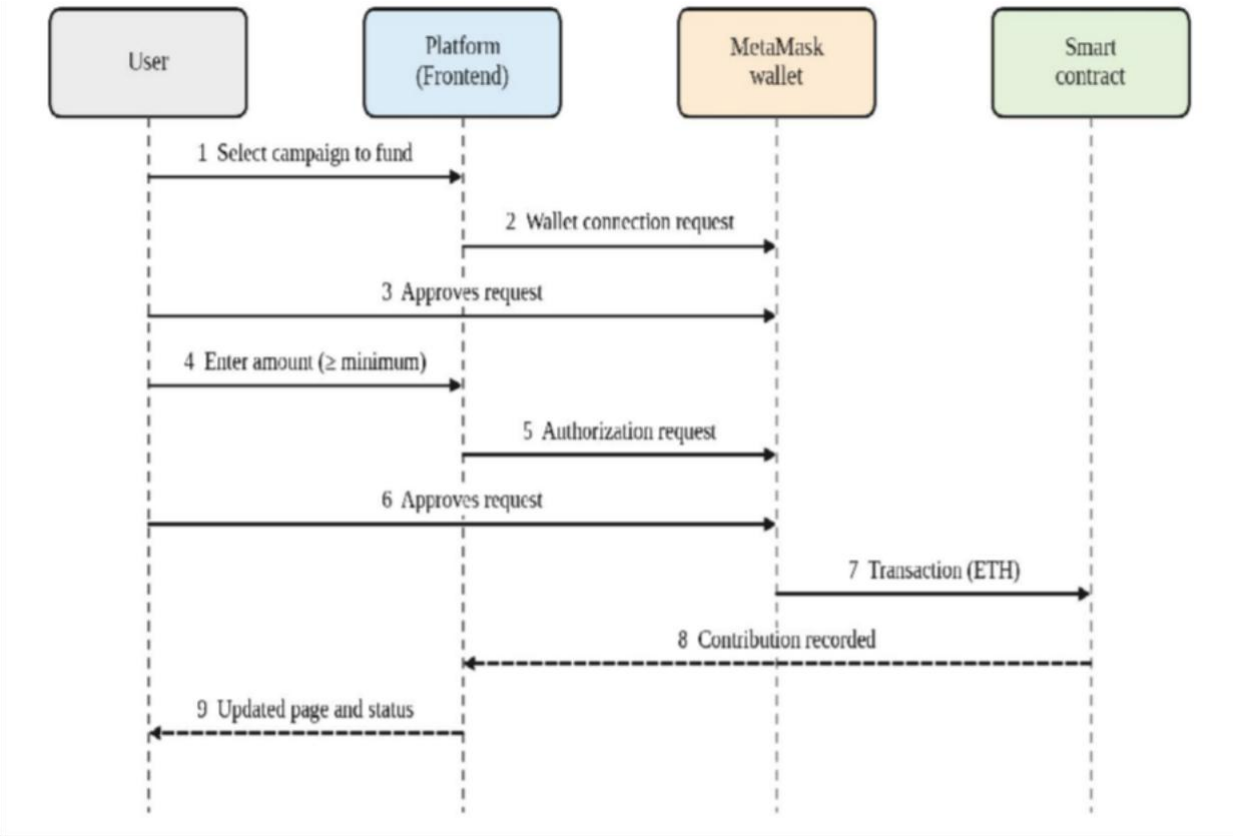


Fig. 5. Sequence of a contribution from campaign selection to the updated page. Dashed arrows are returns

Smart Contract Implementation

The back end is a Solidity contract named Crowdfunding. A Campaign structure holds the owner address, title, description, target, deadline, amount collected, image, the lists of donors and donations, and an active flag; campaigns are kept in a mapping indexed by a running counter. The function createCampaign requires the deadline to lie in the future before it stores the details and returns the new campaign index. The function donateToCampaign accepts Ether only while the campaign is active and before its deadline, appends the donor and the amount, adds the value to the collected total, and marks the campaign inactive once the target has been reached. The function withdrawFunds is intended for the campaign owner only; it requires that funds exist and that the deadline has passed, sets the collected amount to zero before transferring the Ether so that the same funds cannot be withdrawn again, and then marks the campaign finished. Read functions return the list of donors with their amounts and the list of all campaigns.

It should be noted that the contract excerpt included in the project report implements owner-restricted withdrawal after the deadline, whereas the contributor voting and refund behaviour are described in the design documentation, the class diagram (which includes a request class with an approval count and an approve operation) and the test plan. The paper therefore treats voting and refunds as part of the design and the test plan, and the withdrawal path of the excerpt as the part visible in the code.

Frontend Implementation

The interface is written in React.js with TypeScript and styled with Tailwind CSS, and it reaches the contract through a shared state context that exposes the contract functions.

The campaign-creation page collects the creator's name, the campaign title, the story, the goal in Ether, an end date and an image address. Before submission it checks that the image address points to a valid image, converts the goal to the smallest unit of Ether with Ethers.js and shows a loader while the transaction is processed; if the image address is invalid, an error message is displayed and the field is cleared. The campaign-details page shows the days left, the amount raised against the target, the number of backers, the creator's address, the story and the list of donors. It also displays Indian rupee equivalents obtained from a public price service, offers a field for funding the campaign, and shows a withdrawal button only when the connected wallet belongs to the campaign owner.

Technology Stack and Modules

Table III lists the technologies named in the report, and Table IV summarizes the modules together with the evidence for each in the report.

Layer	Technology	Role
User interface	React.js (TypeScript), Tailwind CSS v4.1.7	Campaign pages, forms, and styling
Smart contracts	Solidity v0.8.30	Campaign, contribution, and withdrawal logic
Development environment	Hardhat	Compile, test, and deploy contracts
Blockchain access	Ethers.js; Infura	Interface-to-contract communication; test-network access
Wallet	MetaMask v12.18.2	Authentication and transaction approval
Platform	Ethereum (test network)	Permanent, public record of

Layer	Technology	Role
		transactions
Runtime and tools	Node.js v22.16.0; VS Code; Git	Dependencies, development server, and version control

Module	Main function	Evidence in the report
Wallet connection	Connect MetaMask; authenticate users and approve transactions	Contract function and screens; TC-02 to TC-04
Campaign management	Create campaigns; list campaigns and show details	Screens; test cases TC-01, TC-12
Search	Find campaigns by name or keyword	Screen; TC-05
Contribution	Send Ether to a campaign; record donor and amount	Contract function; TC-06, TC-07
Withdrawal and approval	Creator raises request; contributors vote; majority releases funds	Owner-restricted withdrawal in contract excerpt; voting in design and TC-08 to TC-10
Refund	Contributors reclaim funds if goal is not met	Design and TC-11

TESTING AND RESULTS

Implemented Results

The application was run in a browser against an Ethereum test network, and the screens in Fig. 6 were captured from it. The home page (a) lists the available campaigns as cards, each showing its category, title, a short description, the amount raised, the days left and the creator’s address; at the time of capture, the page showed forty campaigns. The

campaign-details page (b) displays the picture, the days left, the amount raised against the goal of 0.7 ETH with its rupee equivalent, the number of backers, the creator, the story, the donor list and the funding panel with the withdrawal button. The creation form (c) contains the fields described in Section 8.5, filled in with a sample campaign. Finally, the wallet view (d) shows MetaMask connected to the Sepolia test network, with a confirmed “Donate To Campaign” transaction of 0.01 test Ether and a “Create Campaign” transaction, which confirms that creation and contribution reach the chain through the wallet. The design text names the Goerli test network, whereas this screenshot shows Sepolia; the report does not explain the difference, so both are mentioned here.

Functional Testing

Fourteen test cases were defined and every one is recorded as passed in the report; Table V groups them by area. The cases cover wallet connection, campaign creation with valid and missing details, display of campaigns, keyword search, contribution with a valid and an insufficient amount, the withdrawal request, voting, release of funds after majority approval, refunds for an expired campaign that missed its goal, rejection of transactions without a connected wallet, loading of 500 campaigns, and responsive display on a mobile device. The performance case expects the 500 campaigns to load within an acceptable time, given as under three seconds as an example; the report does not give any measured timing.

TABLE V: TESTING SUMMARY (TEST CASES AS RECORDED IN THE REPORT)

Area	Cases	Scenario and expected behavior	Status
Wallet and	TC-01,	Wallet connects and shows the address; contributing	Pass

Area	Cases	Scenario and expected behavior	Status
security	TC-12	without a wallet shows “Please connect wallet.”	
Campaign creation	TC-02, TC-03	Valid details create a campaign stored on the blockchain; an empty title or goal shows an error.	Pass
Viewing and search	TC-04, TC-05	Campaigns are retrieved from the blockchain and displayed; keyword “Education” returns matching campaigns.	Pass
Contribution	TC-06, TC-07	A 1 ETH contribution updates the amount raised; a 10 ETH contribution from a wallet holding 5 ETH is rejected with an error.	Pass
Withdrawal and voting	TC-08 to TC-10	Request is stored and pending; vote is recorded; funds are released after majority approval.	Pass
Refund	TC-11	Contributors can withdraw after expiry if the goal is not reached.	Pass
Performance	TC-13	500 campaigns load within acceptable time.	Pass
Usability	TC-14	Layout adapts to a mobile screen without overlap.	Pass

Expected Benefits

The following benefits are expected consequences of the design and have not been measured. Removing the intermediary should lower service charges. Public, immutable records should make contributions and withdrawals easier to verify. Voting-based release should reduce the risk of misuse, and automatic refunds should protect contributors when a goal is missed. Worldwide participation through cryptocurrency should widen access. Confirming these effects would require a trial with real users and a comparison of fees, gas costs and user experience, which the project does not report.

ADVANTAGES AND LIMITATIONS

The main advantage of the system is that trust is placed in transparent code instead of a company. Contributors can see where their money went, creators keep the full amount they raise apart from network fees, and the contributors who funded a campaign take part in decisions about its money. The modular structure, with a React.js interface, a Solidity contract and a wallet-based login, also makes the system straightforward to extend.

There are limitations as well. Participants need a cryptocurrency wallet and some familiarity with it, and gas fees are payable on every transaction, points that the literature also raises [2, 3, 8]. Scalability and regulatory questions remain open. The evaluation reported here is functional testing on a test network; no formal security audit, gas analysis or load measurement is described. The contract excerpt does not yet contain the voting and refund logic that the design specifies, so bringing the deployed contract fully in line with the design is the most immediate step. Finally, the campaign image is supplied as an address on the web, and exchange-rate figures depend on an external service.

FUTURE ENHANCEMENTS

Several extensions are suggested in the report. Multi-currency support could allow contributions in other cryptocurrencies such as Bitcoin, USDT, BNB and MATIC. Fiat payment integration could allow people with no crypto experience to contribute by card, UPI or bank transfer, which would address the usability gap noted in Table II. Cross-chain compatibility, for example across Ethereum, Binance Smart Chain, Polygon and Solana, is expected to reduce gas fees and improve efficiency. AI-based recommendations could suggest campaigns according to a contributor’s interests, past

contributions and trending causes. A mobile application could be combined with stronger security features such as two-factor authentication, biometric verification and fraud detection. The user manual also mentions IPFS as an optional store for campaign documents and media.

CONCLUSION

This paper has presented a blockchain-based crowdfunding platform in which smart contracts, not an intermediary, hold campaign rules and funds. The design brings together campaign creation, wallet-based contribution, public recording of transactions, majority approval of withdrawals and refunds when a goal is missed. The implementation uses React.js, Tailwind CSS, Solidity, Hardhat, Ethers.js and MetaMask, and its working screens and fourteen recorded test cases show that the core workflow functions on an Ethereum test network.

The platform is best understood as a working prototype and a step toward more transparent crowdfunding. Its main strengths are transparency, immutability and the removal of intermediaries; its main limits are the learning curve of cryptocurrency, transaction costs and the absence of measured performance and security results. Future work should align the deployed contract with the full voting and refund design, evaluate cost and security, and extend the platform to other currencies, chains and payment methods.

Author Information

Guna Sagar R

Master of Computer Applications (MCA)

S.E.A. College of Science, Commerce and Arts

K. R. Puram, Bangalore, India

References

Chattani, A., Sharma, A., & Manhar, A. (2023). Research paper on crowdfunding using blockchain. Journal of Emerging Technologies and Innovative Research, 10(6), 518–524.

Fernandes, F., Gharat, H., Kadam, A., & Kamil, A. (2023). Crowd funding platform using blockchain. International Journal of Innovative Research in Technology, 9(10).

Garg, N., Seth, S., Rastogi, N., Kumar, R., Gupta, V., & Rawat, S. S. (2022). Crowdfunding using blockchain technology: A review. Global Journal of Innovation and Emerging Technology, 1(2), 8–14.

Guggenberger, T., Schellinger, B., von Wachter, V., & Urbach, N. (2024). Kickstarting blockchain: Designing blockchain-based tokens for equity crowdfunding. Electronic Commerce Research, 24, 239–273.

Gupta, P., Sangle, V., Patil, R., Jadhav, A., Aher, M., & Khivansara, S. (2024). Secure way of crowdfunding using blockchain. International Journal of Research Publication and Reviews, 5(5), 2092–2098.

Gupta, S., Gupta, R., Gangwar, L., Gupta, A., & Shinde, V. (2024). Easy-starter: An online crowdfunding website using blockchain. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 10(2), 232–241.

Saadat, M. N., Halim, S. A., Osman, H., Nassr, R. M., & Zuhairi, M. F. (2019). Blockchain-based crowdfunding systems. Indonesian Journal of Electrical Engineering and Computer Science, 15(1), 409–413.

Tiganoaia, B., & Alexandru, G.-M. (2023). Building a blockchain-based decentralized crowdfunding platform for social and educational causes in the context of sustainable development. Sustainability, 15, Article 16205.

VP, S., Krishna, H., Jain, L., & Ahmed, H. (2023). Crowd-funding using blockchain. In Proceedings of the 2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing, and Automation (ICAECA).

Source Basis

This research paper was prepared from the supplied “GUNA_FULL_PJ_A4(1).pdf” project report. The structure, typography, alignment, section treatment, and table presentation were modeled on the supplied “research paper tms.pdf” reference. No external research was added to the substantive project content.

Formatting note: This version intentionally contains no running header, footer, page number, or other recurring page element, as requested.