

Decentralized Voting System Using Blockchain and Web3

K J. Rajashekar, C. Divya, M H. Bhavyashree, D S. Chinmayi, N. Chithritha, M R. Vidyashree, C. Corresponding author: Divya

Department of Information Science and Engineering, Kalpataru Institute of Technology, Tiptur, Tumkur, Karnataka, India

Published October 2026

Abstract

Conventional electronic voting depends on a central server and database, so voters and observers must trust the organization that operates them. This paper presents a decentralized voting application in which the main election rules are enforced by a Solidity smart contract running on an Ethereum-compatible blockchain. A React and Next.js front end communicates with the contract through Web3 and an Ethereum wallet. The organizer, who deploys the contract, registers candidates and voters and activates or deactivates voting. Before a vote is accepted, the contract checks that voting is active, that the sender is a registered voter, that the voter has not voted before, and that the chosen candidate exists. Accepted votes update the candidate's count and emit events that the interface uses to refresh its data. The system is built with open-source tools (Hardhat, OpenZeppelin, Ethers, Next.js) and is presented as a working prototype. The paper describes the architecture, data structures, modules and test cases, shows the user-interface screens, and discusses limitations such as voter authentication, privacy, scalability and the absence of an automated test suite for the voting contract.

Keywords: blockchain; decentralized application; smart contract; Web3; electronic voting

1 Introduction

Elections and polls are only useful when participants trust how votes are collected and counted. In most electronic voting systems the voter registry, the ballot data and the tally sit in a database that is controlled by one organization. Any error or misuse in that database is hard for an outsider to detect, and users must simply believe the operator. The aim of this work is to reduce that dependence by moving the core voting rules into a program that runs on a blockchain, where its code and its recorded actions can be inspected.

1.1 Blockchain and smart contracts

A blockchain is a ledger shared across many computers, where each new record is chained to the earlier ones and is hard to alter. Smart contracts are programs stored on this ledger that run automatically when a transaction calls them. Their rules are public and execute the same way for everyone, which suits voting, where fairness must be visible.

1.2 Requirements of an electronic voting system

For voting to be trusted, the system must allow only eligible people to vote, count each person once, store votes accurately, stay open to inspection, and reserve administrative actions for the organizer. Our prototype builds these rules directly into its contract code.

1.3 Decentralized application (DApp)

A DApp moves the key operations of an application onto a smart contract. Ours combines a web interface, a wallet, a Solidity contract and Web3 calls between the interface and the blockchain, with blockchain transactions for every voting operation. Its objectives are to (i) implement voting logic in a smart contract, (ii) let the organizer register candidates and voters, (iii) allow only registered voters to vote once, (iv) maintain candidate vote counts, and (v) let the organizer start or stop voting.

2 Literature survey

Many researchers have applied blockchain to voting. Some works focus on the basic idea of replacing a central tally with a shared ledger. Study [1] presents blockchain as a technology that has moved beyond cryptocurrency into online services such as voting, and [2] proposes a platform that uses smart contracts and flexible consensus to give voters and officials a transparent channel. Work [3] builds on Multichain in response to falling public confidence in elections, while [4] targets small elections such as board meetings using Ethereum, Truffle, Ganache and MetaMask.

A second group addresses specific technical problems. Authentication is handled with ECC and SHA-256 in [5], which also lets voters revise their vote before a deadline, and with biometric-derived keys in [9], which preserves anonymity. Study [6] uses Hyperledger, and [7] introduces intelligent agents for auditability and end-to-end verification. For privacy and scalability, [8] relies on off-chain zero-knowledge computation, and [10] uses a cloud-based hybrid blockchain with digital signatures and PBFT consensus. Country-level studies in [11] (Morocco, Solana) and [16] (Indonesia) emphasize transparency and auditability.

The most recent studies concentrate on security and efficiency. Work [12] surveys architectures, trends and open challenges. Study [13] examines coercion resistance, [14] looks at counting performance, [17] proposes delegated proof-of-stake for scalability, and [18] a hybrid cryptographic framework that lowers energy use. Work [15], the closest to ours, offers an Ethereum mechanism with role-based access, candidate management and duplicate-vote prevention. Two further Ethereum prototypes are [19], which uses Truffle, Ganache, React and MetaMask in a mock election, and [20], which provides administrator and voter modules.

Overall, these studies agree that blockchain improves transparency and resists tampering, yet authentication, privacy and scalability remain unresolved. We contribute a compact, fully documented DApp built with current open-source tools, and we state its limitations openly.

3 Methods

The system is designed as a layered DApp. Figure 1 shows how the user, interface, Web3 layer, wallet, contract and blockchain interact.

3.1 System architecture

User and front end. The user opens the application in a browser. The front end, written with React and Next.js, provides a home page, navigation, candidate registration, voter registration, a voter list and the voting interaction. **Web3 layer.** Web3 requests access to Ethereum accounts, reads the connected wallet address, creates a contract instance, calls contract functions, sends transactions and listens for blockchain events. **Wallet.** The Ethereum-compatible browser wallet supplies the account that signs transactions. **Smart contract and blockchain.** The Solidity contract keeps candidates,

voters, votes, voting status and access control, and the blockchain stores its state and processes transactions.

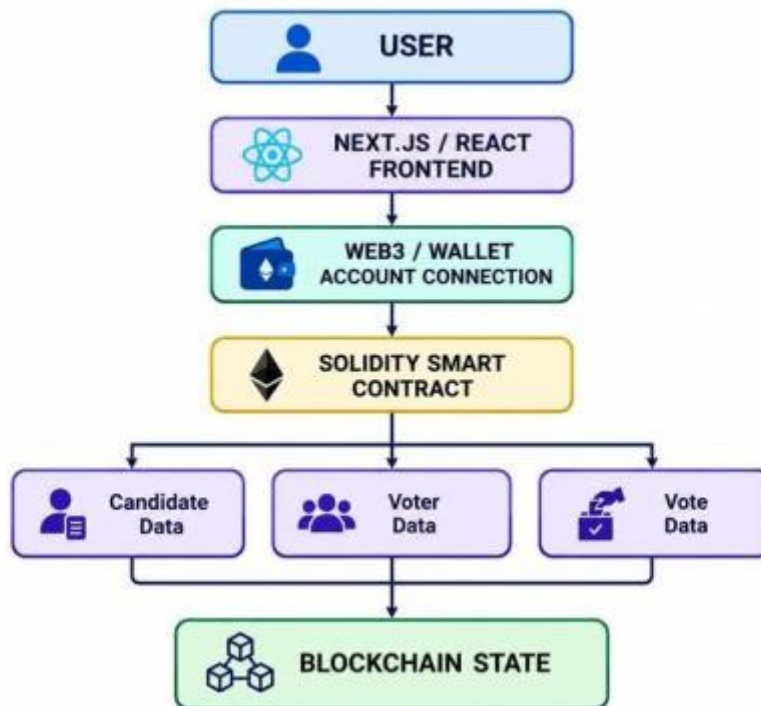


Figure 1. System architecture of the blockchain-based voting system.

3.2 Workflow

The organizer deploys and controls the contract, registers voters and adds candidates. A user connects a wallet, and the application checks the user's status. A registered voter selects a candidate and submits a vote. The contract verifies that the voter has not already voted, increases the selected candidate's count and emits an event. The organizer can switch voting between active and inactive at any time. Figure 2 summarizes the activity flow.

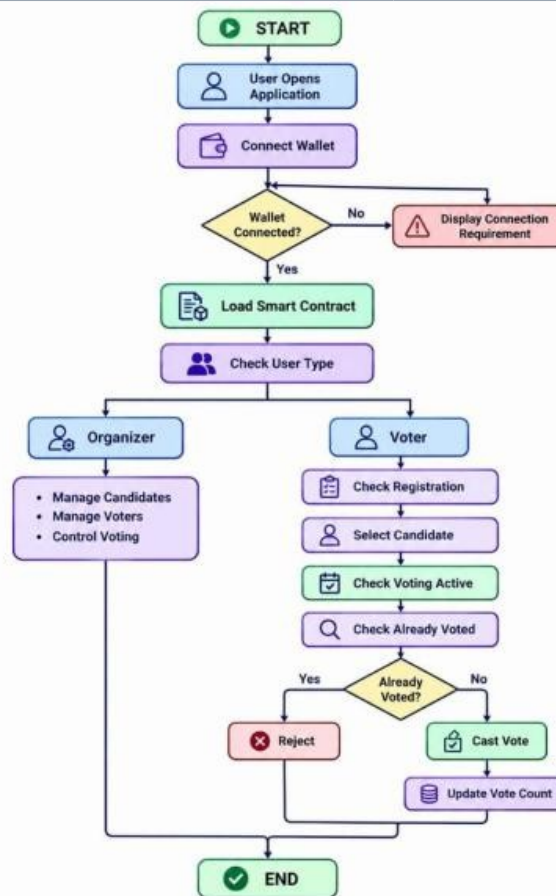


Figure 2. Activity flow of the voting application.

3.3 Smart contract design

The main contract, named *Create*, targets Solidity ^0.8.9 and imports the OpenZeppelin *Counters* utility, which generates incrementing candidate and voter identifiers. A *Candidate* structure holds the candidate ID, age, name, vote count, the creating blockchain address and an IPFS field. A *Voter* structure holds the voter ID, name, wallet address, a *hasVoted* flag, the voted candidate ID and an *isRegistered* flag. Three mappings link candidate ID to candidate details, wallet address to voter details, and candidate ID to the list of voter addresses that voted for it.

Two modifiers protect the functions. *onlyOrganizer* limits candidate and voter registration to the organizer, which is initialised as the address that deploys the contract. *votingIsActive* allows an operation only while voting is active; a companion *votingIsInactive* modifier is also defined. The contract emits four events: *CandidateCreate*, *VoterRegistered*, *VoteCast* and *VotingStatusChanged*. The front end listens for them and reloads its data. The contract operations and their guards are summarized in Table 1.

Table 1. Contract operations and their guards.

Operation	Who may call	Checks performed
Add candidate	Organizer	Organizer access; voting active
Register voter	Organizer	Organizer access; address not already registered
Cast vote	Registered voter	Voting active; registered; not voted; candidate exists

Operation	Who may call	Checks performed
Toggle voting	Organizer	Organizer access
Read data	Anyone	Candidate, voter, count queries

3.4 Implementation modules

Wallet connection: the application detects an Ethereum provider, requests account access and stores the connected address in its state. **Contract initialization:** a Web3 instance is created, the account is loaded, and the contract ABI and deployed address are used to create the contract object. **Organizer verification:** the organizer address is read from the contract and compared with the connected account; a match unlocks administrative pages. **Candidate registration:** a form collects name, wallet address, age and an IPFS field; the current prototype fills the IPFS field with a placeholder value instead of uploading a file. **Voter registration:** the organizer enters a name and blockchain address, and the contract rejects an address that is already registered before assigning a new voter ID. **Voting:** the voter submits a candidate ID, and after the four checks the contract marks the voter as voted, stores the candidate ID, increments the count, appends the voter to the candidate's list and emits *VoteCast*. **Voting status management:** the organizer toggles voting between active and inactive. **Data retrieval:** read functions return one or all candidates, the candidate total, voter details, registration and voted status, and the voter total.

3.5 Software environment

The contract is written in Solidity and developed with Hardhat, which provides compilation, configuration, deployment scripts and test files. OpenZeppelin supplies the counters, Ethers is included in the tooling, and the interface uses Next.js routing, React state, effects and context. Open-source tools keep the prototype inexpensive. A modest development machine (for example Core i3 class, 4 GB RAM) and a browser with a wallet extension are sufficient.

3.6 Test cases

Table 2 lists the functional test cases defined for the contract and the behaviour the contract logic enforces. They were prepared as manual system tests.

Table 2. Functional test cases.

Test case	Expected result
Organizer registers candidate	Candidate added
New voter registered	Marked registered; duplicate rejected
Registered voter votes for valid candidate	Vote recorded; count increases
Unregistered wallet votes	Transaction rejected
Registered voter votes twice	Second vote rejected
Vote for non-existent candidate	Transaction rejected
Vote while voting inactive	Vote not accepted

4 Results

The prototype was run in a browser with a wallet and produced the screens shown below. Figure 3 shows the candidate registration page where the organizer enters name, address and age, and Figure 4 shows the candidate list with each candidate's voting status. Figure 5 shows the live results panel.

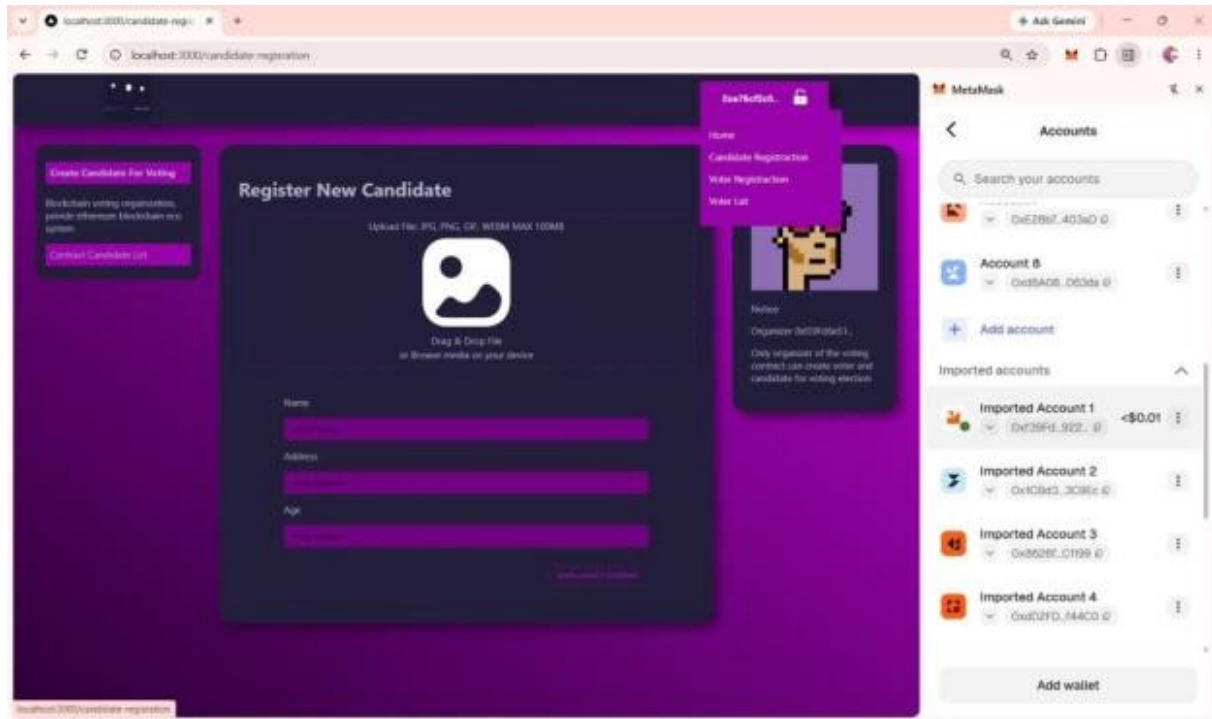


Figure 3. Candidate registration page.

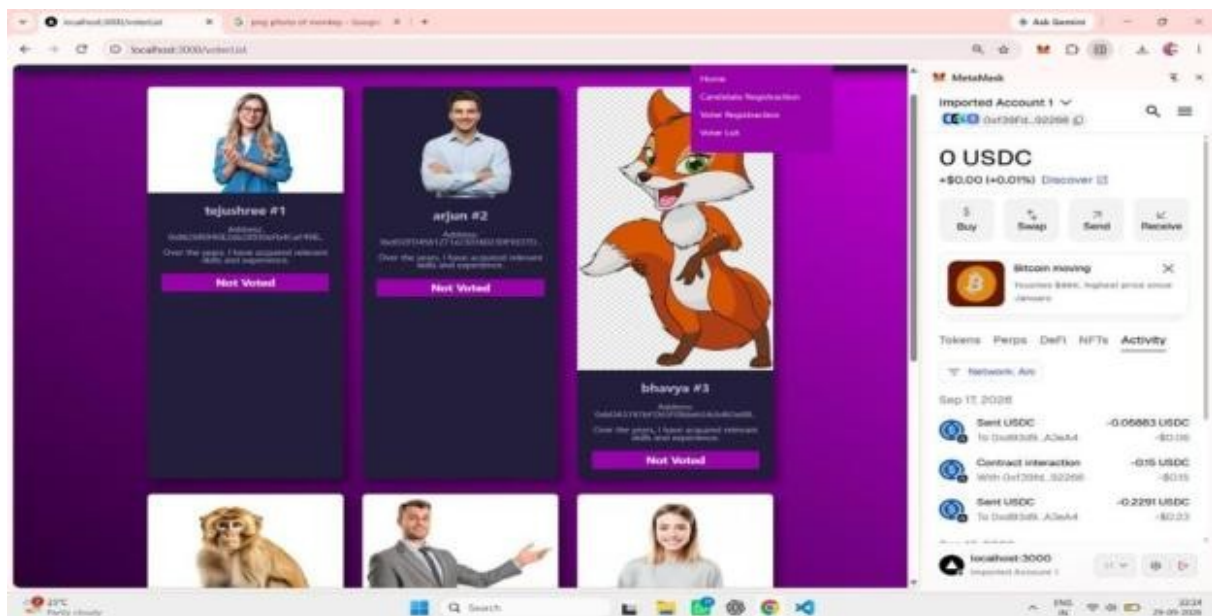


Figure 4. Candidate list and voting page.

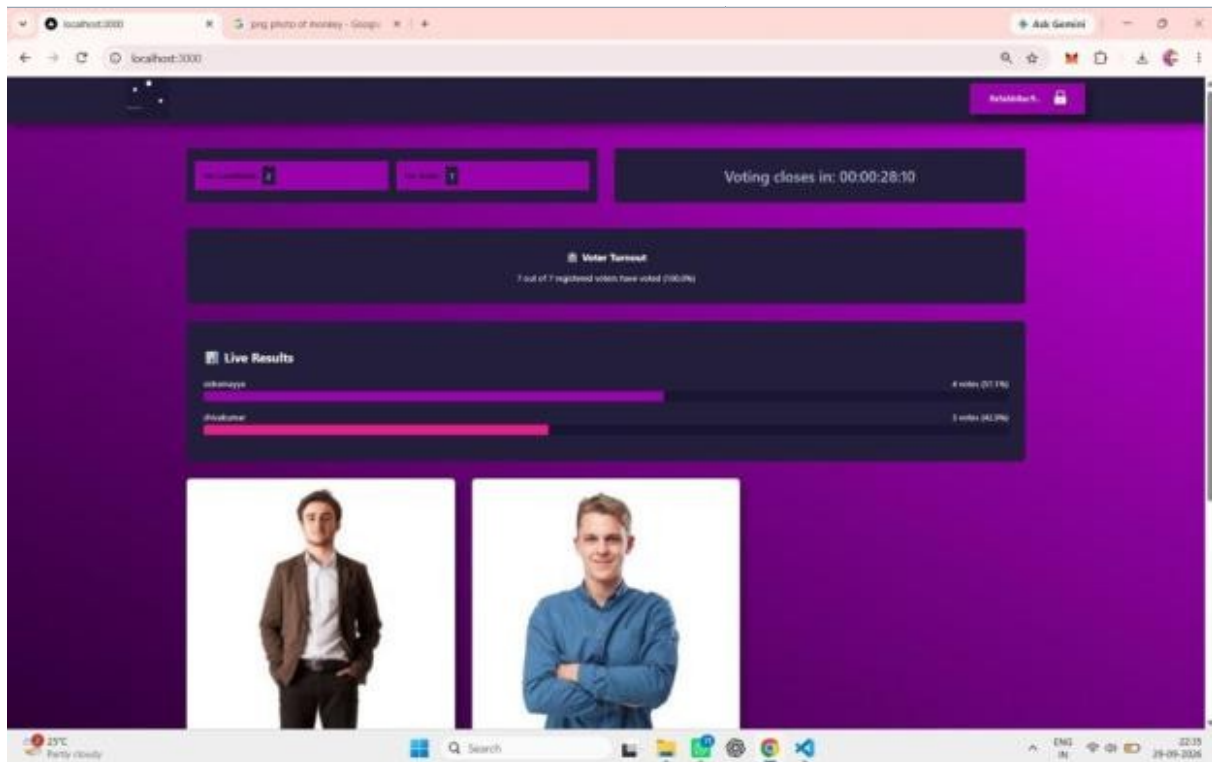


Figure 5. Live voting results.

5 Discussion

The design places the key rules inside the contract, so a vote cannot be accepted unless voting is active, the sender is registered, the sender has not voted and the candidate exists. Organizer-only registration prevents ordinary users from changing the candidate or voter lists, and events keep the interface in step with the ledger. This gives transparency, because the rules are public, and reduces reliance on a central database, which supports the motivation in [1–4] and matches the access-control and double-voting focus of [15].

5.1 Limitations

Several limits should be noted. First, the project test folder contains only the default Hardhat *Lock* test, so an automated test suite for the voting contract is not yet available; the cases in Table 2 are manual. Second, the IPFS field uses a placeholder rather than a real upload. Third, identity is bound only to a wallet address chosen by the organizer, so strong voter authentication, ballot secrecy and coercion resistance are not provided, in contrast to [8], [9] and [13]. Fourth, vote records linked to addresses and the per-candidate voter list may expose voting choices. Finally, scalability and cost on a public network were not measured, and no quantitative performance results are reported.

6 Conclusion

This paper presented a decentralized electronic voting prototype built with a Solidity smart contract, a React and Next.js interface and Web3 wallet integration. The contract supports candidate and voter registration, vote casting, vote counting and voting status control, and it validates organizer rights, voter registration, duplicate voting, candidate existence and voting status. The prototype demon-

strates how blockchain transactions can carry the core voting rules and provide transparency and tamper resistance.

Future work will add an automated test suite for the contract, genuine IPFS storage, stronger voter authentication such as multi-factor checks or decentralized identity, privacy techniques such as zero-knowledge proofs, and Layer-2 scaling. Further improvements include a mobile-friendly interface and transparent audit tracking.

References

- [1] Jangada A, Dadlani N, Raina S, Sooraj V, Buchade AR. De-centralize voting system using blockchain. In: *2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*. Pune, India; 2022. p. 1–5.
- [2] Jadhav J, Ghude P, Pashte S, Sonawane A. Decentralized based e-voting system using blockchain technology. In: *2024 International Conference on Advances in Computing Research on Science Engineering and Technology (ACROSET)*. Indore, India; 2024. p. 1–6.
- [3] Balti A, Prabhu A, Shahi S, Dahifale S, Maheta V. A decentralized and immutable e-voting system using blockchain. In: *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*. Coimbatore, India; 2023. p. 1434–1439.
- [4] Patidar K, Jain S. Decentralized e-voting portal using blockchain. In: *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. Kanpur, India; 2019. p. 1–4.
- [5] Al-Madani M, et al. *Securing e-voting based on blockchain in P2P network*. EURASIP Journal on Wireless Communications and Networking. 2019.
- [6] The decentralized voting model using the Hyperledger platform. In: *2019 International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo)*. 2019. <https://doi.org/10.1109/UkrMiCo47782.2019.9165368>
- [7] Towards the intelligent agents for blockchain e-voting system. *Procedia Computer Science*. 2018;141:239–246. <https://doi.org/10.1016/j.procs.2018.10.177>
- [8] A scalable decentralized privacy-preserving e-voting system based on zero-knowledge off-chain computations. *Journal of Information Security and Applications*. 2023;79:103645. <https://doi.org/10.1016/j.jisa.2023.103645>
- [9] A biometrics-generated private/public key cryptography for a blockchain-based e-voting system. *Egyptian Informatics Journal*. 2024;25:100447. <https://doi.org/10.1016/j.eij.2024.100447>
- [10] E-voting system using cloud-based hybrid blockchain technology. *Journal of Safety Science and Resilience*. 2024;5(1):102–109. <https://doi.org/10.1016/j.jnlssr.2024.01.002>
- [11] Blockchain-based electronic voting systems: A case study in Morocco. *International Journal of Intelligent Networks*. 2024;5:38–48. <https://doi.org/10.1016/j.ijin.2024.01.004>
- [12] Blockchain for securing electronic voting systems: A survey of architectures, trends, solutions, and challenges. *Cluster Computing*. 2024. <https://doi.org/10.1007/s10586-024-04709-8>
- [13] Efficient, usable and coercion-resistant blockchain-based e-voting. *Journal of Information Security and Applications*. 2025;92:104074. <https://doi.org/10.1016/j.jisa.2025.104074>
- [14] Leveraging blockchain for robust and transparent e-voting systems. *Computer Science & Engineering*. 2025;3:100086. <https://doi.org/10.1016/j.csa.2025.100086>
- [15] A secure approach to mitigate double voting in electronic voting systems using blockchain technology. In: *2025 8th International Conference on Circuit, Power & Computing Technologies (ICCPCT)*. 2025. <https://doi.org/10.1109/ICCPCT65132.2025.11176586>
- [16] Implementation of blockchain-based electronic voting system: A case study in Indonesia. *Procedia Computer Science*. 2025;269:1–12. <https://doi.org/10.1016/j.procs.2025.08.253>
- [17] Blockchain-enabled smart contracts and prioritized delegated proof-of-stake paradigm for secure and scalable electronic voting systems. *Blockchain: Research and Applications*. 2026;7(4):100348. <https://doi.org/10.1016/j.bcr.2025.100348>
- [18] A sustainable hybrid cryptographic framework for energy optimization and scalability in blockchain-based e-voting. *Scientific African*. 2026;31:e03191. <https://doi.org/10.1016/j.sciaf.2026.e03191>

-
- [19] Pradhan S, Sanjta A, Reddy MDK, Raviteja N, Kumar TD. Decentralized voting system using blockchain technology. Lovely Professional University; SSRN. <https://ssrn.com/abstract=4483688>
- [20] Reddy KK, Kumar GV, Sirimulla SS, Singh C, Reddy CK, Reddy PPS. Decentralized voting system using blockchain. *International Journal of Scientific Research in Science, Engineering and Technology*. 2024;11(2):211–219. <https://doi.org/10.32628/IJSRSET2310216>