IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, June 2026
Impact Factor: 8.2

Blockchain Based Secure E-Voting System

Prof. Jayshri Mankar[1], Dilip Maurya[2], Utkarsh Karpe[3], Nilesh Patil[4], Krushna Patil[5]
[*2,3,4,5]Student, Department of Computer Engineering, Alard College of Engineering and Management, Pune, India.
[*1]Assistant Professor, Department of Computer Engineering, Alard College of Engineering and Management, Pune

Abstract: *The proliferation of digital infrastructure has created an urgent need to modernize traditional electoral processes while preserving the core democratic principles of transparency, fairness, and voter privacy. Conventional electronic voting systems, although faster than paper ballots, remain vulnerable to single-point-of-failure risks, unauthorized data tampering, and a general lack of verifiability that erodes public trust. This paper presents the design and implementation of a Blockchain Based Secure E-Voting System that combines the flexibility of the MERN stack (MongoDB, Express.js, React.js, Node.js) with the immutability and decentralization guarantees of an Ethereum-based blockchain layer. In the proposed architecture, the MERN stack governs application logic, user interface rendering, voter registration, and administrative workflows, while every vote cast by an authenticated voter is recorded as a transaction on a permissioned Ethereum blockchain through a dedicated smart contract. This hybrid design ensures that once a vote is cast, it cannot be altered, deleted, or duplicated, since each transaction is cryptographically linked to the preceding block through SHA-256 hashing. The system further incorporates JWT-based authentication, bcrypt password hashing, SendGrid email OTP verification, Aadhar-based identity validation, and role-based access control to separate Voter and Admin privileges. A working prototype was implemented and deployed, with the frontend hosted on Vercel and the backend on Render, backed by MongoDB Atlas for off-chain data and a blockchain layer for vote-integrity assurance. Experimental evaluation across more than twenty-five REST API endpoints demonstrated reliable performance, successful duplicate-vote prevention, and consistent vote tallying. The results indicate that integrating blockchain as a security and verification layer atop a conventional web stack offers a practical, scalable, and trustworthy pathway toward next-generation digital elections.*

Keywords: Blockchain, E-Voting, Ethereum, Smart Contract, MERN Stack, Decentralization, Cybersecurity, JWT Authentication

I. INTRODUCTION

Elections form the structural backbone of democratic governance, and the integrity of the voting process directly determines the legitimacy of the resulting government. For decades, paper-based ballots remained the dominant mechanism for casting and counting votes, but the process is slow, labour-intensive, and prone to human error during manual counting. The advent of Electronic Voting Machines (EVMs) and web-based voting portals addressed speed and convenience, yet introduced new categories of risk: centralized servers can be attacked, databases can be altered by privileged insiders, and voters have no independent way of verifying that their vote was recorded as cast.

Blockchain technology, originally developed as the underlying ledger for cryptocurrencies, offers a compelling solution to these problems because of three intrinsic properties: immutability, decentralization, and transparency. Once a transaction is appended to a blockchain, it cannot be silently modified without invalidating every subsequent block in the chain. When this property is applied to electronic voting, each vote becomes a permanent, tamper-evident record that can be independently audited without compromising voter anonymity. This paper proposes a Blockchain Based Secure E-Voting System that layers an Ethereum-based blockchain on top of a conventional MERN stack web application. The web layer handles the day-to-day operational requirements of an election — voter registration, identity document verification, candidate management, and result visualization — while the blockchain layer is reserved exclusively for the security-critical operation of vote casting and vote storage. By restricting blockchain usage to the

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, June 2026
Impact Factor: 8.2

vote transaction itself, the system avoids the performance overhead of running an entire application on-chain while still gaining the tamper-resistance benefits of distributed ledger technology. The remainder of this paper is organized as follows. Section II reviews related work in electronic and blockchain-based voting systems. Section III describes the proposed system architecture. Section IV details the methodology and implementation, including the smart contract design and database schema. Section V presents the API design. Section VI discusses results and performance evaluation. Section VII offers a comparative analysis against traditional e-voting systems. Section VIII concludes the paper and outlines directions for future work.

II. LITERATURE REVIEW

Several researchers have explored the application of distributed ledger technology to electronic voting in recent years. Early electronic voting research focused primarily on cryptographic protocols such as homomorphic encryption and mix-networks to preserve voter privacy while allowing public verifiability of the final tally. While mathematically robust, these approaches often required significant computational overhead and were difficult to deploy in real-world elections with non-technical voters. With the emergence of blockchain platforms such as Bitcoin and Ethereum, researchers began investigating distributed ledgers as a more practical substrate for verifiable voting. Studies have shown that recording votes as blockchain transactions allows any voter to verify, using a public transaction identifier, that their vote was included in the final count without revealing which candidate they selected.

Ethereum ' s smart contract capability, in particular, has been widely studied because it allows the rules of an election — eligibility checks, vote-counting logic, and result declaration — to be encoded directly into self-executing code that runs identically across every node in the network. Other work has focused on hybrid architectures that combine an off-chain web application with an on-chain ledger, citing concerns about the transaction throughput and gas costs associated with running an entire application directly on a public blockchain. These hybrid designs typically use a conventional relational or NoSQL database for non-sensitive operational data — such as candidate profiles, election schedules, and user interface state — while reserving the blockchain exclusively for the vote transaction itself.

This division of responsibility is consistent with the architecture proposed in this paper, in which MongoDB stores operational data and the Ethereum layer stores the cryptographically verifiable vote record. A recurring theme across the literature is the importance of combining blockchain-based vote integrity with strong identity verification at the point of registration, since a blockchain can guarantee that a recorded vote is unaltered, but it cannot, by itself, guarantee that the person casting the vote is who they claim to be. Consequently, several proposed systems pair blockchain vote storage with traditional authentication mechanisms such as government-issued identity document verification, one-time password (OTP) confirmation, and multi-factor authentication.

The system proposed in this paper follows this established pattern by integrating Aadhar document verification, email-based OTP confirmation, and JWT-secured sessions alongside the blockchain vote layer, thereby addressing both the integrity of the vote and the authenticity of the voter. Existing research gaps that this paper aims to address include: (i) the lack of openly documented, end-to-end implementations that combine a modern JavaScript-based web stack with an Ethereum smart contract for vote storage; (ii) limited discussion of how role-based administrative workflows — such as candidate management and result declaration — can be cleanly separated from the immutable vote ledger; and (iii) insufficient empirical performance data on API-level testing for such hybrid systems. The present work contributes a fully implemented and deployed prototype, evaluated across twenty-five-plus REST endpoints, to help close these gaps.

III. METHODOLOGY

The proposed system follows a three-tier architecture in which a conventional MERN-based web tier handles application logic and user experience, while a dedicated blockchain security tier ensures that every cast vote is recorded immutably and verifiably. Figure 1 illustrates the high-level architecture of the system.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, June 2026
Impact Factor: 8.2

A. Architectural Overview

The Client Tier is built using React.js with Vite as the build tool and Tailwind CSS for styling, and is responsible for rendering the Voter Portal and Admin Panel. The Application Tier, built on Node.js and Express.js, exposes a REST API that handles authentication, validation, and orchestration between the operational database and the blockchain layer. The Data Tier is split into two complementary stores: MongoDB Atlas, which persists operational and non-sensitive data such as user profiles, election metadata, and candidate records; and the Ethereum blockchain, which persists only the cryptographically signed vote transactions. This separation ensures that the performance-sensitive, high-frequency operations (browsing elections, viewing candidate details, dashboard analytics) are served quickly from MongoDB, while the security-critical, low-frequency operation (casting a vote) is anchored to the blockchain fortamper-evidence.

Figure 1 — High-level System Architecture

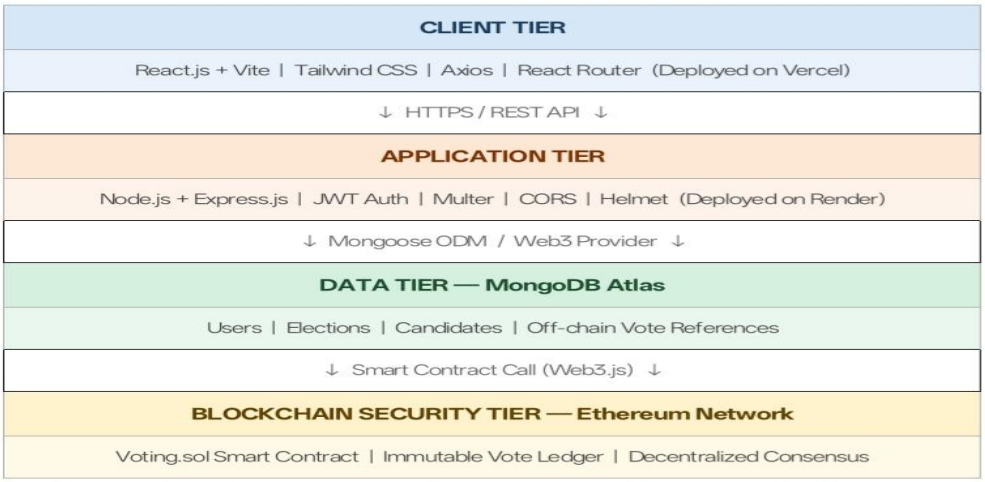

Fig. 1. High-level architecture showing the Client Tier, Application Tier, the MongoDB off-chain data store, and the Ethereum blockchain security layer connected through a smart contract interface.

B. Request And Vote Flow

The end-to-end flow for a typical voting transaction proceeds as follows: (1) the voter authenticates through the React frontend, which sends credentials to the Express backend over HTTPS; (2) the backend validates the JWT token and confirms that the voter has been verified and is eligible for the active election; (3) upon vote submission, the backend constructs a transaction payload containing the election identifier, an anonymized voter token, and the selected candidate identifier; (4) this payload is signed and submitted to the deployed Voting smart contract on the Ethereum network; (5) the smart contract validates that the voter token has not previously voted in this election, records the vote as an immutable transaction, and emits a VoteCast event; (6) the backend listens for contract events and updates the off-chain MongoDB collection with a reference to the on-chain transaction hash for fast retrieval and dashboard reporting; (7) the updated vote count is reflected in real time on the Admin dashboard and, afterresult declaration, on the public results page.

C. Technology Stack

Table I, Table II, and Table III summarize the frontend, backend, and blockchain technologies respectively used to build the system.

DOI: 10.48175/568

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, June 2026
Impact Factor: 8.2

TABLE I. FRONTEND TECHNOLOGY STACK

Technology	Version	Purpose
React.js	18.2.0	UI Framework
Vite	7.3.1	Build Tool
Tailwind CSS	3.4.19	Styling
Framer Motion	12.34.3	Animations
React Router DOM	6.30.3	Client-side Routing
Axios	1.13.5	HTTP Client
React Hot Toast	2.6.0	Notifications
React Icons	5.5.0	Icon Library

TABLE II. BACKEND TECHNOLOGY STACK

Technology	Version	Purpose
Node.js	22.x	Runtime Environment
Express.js	4.x	Web Framework
MongoDB	6.0	Off-chain NoSQL Database
Mongoose	8.x	ODM Layer
JSON Web Token (JWT)	9.x	Authentication
Bcryptjs	2.x	Password Hashing
Multer	1.x	File Uploads
SendGrid	8.x	Email OTP Service
Express Validator	7.x	Input Validation

TABLE III. BLOCKCHAIN SECURITY LAYER

Technology	Version	Purpose
Ethereum	—	Decentralized Blockchain Network
Solidity	0.8.x	Smart Contract Language
Web3.js / Ethers.js	Latest	Blockchain Interaction Layer
Smart Contract (Voting.sol)	Custom	On-chain Vote Recording & Validation

MODELING AND ANALYSIS

This section describes the implementation methodology adopted to build the proposed system, covering authentication and security, the database schema used for off-chain data, and the design of the Ethereum smart contract used for on-chain vote storage.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, June 2026
Impact Factor: 8.2

A. Authentication and Security Layer

Voter registration requires submission of full name, date of birth, residential address, and a scanned Aadhar document in JPG, PNG, or PDF format. The backend validates the applicant' s age to confirm 18+ eligibility before allowing registration to proceed. Upon submission, an Admin reviews the uploaded document and either approves or rejects the registration through the Admin Panel. Approved voters receive an email containing a one-time password (OTP) generated and dispatched through SendGrid, which must be entered to confirm the email address before the account is activated. All passwords are hashed using bcrypt prior to storage, and all authenticated sessions are protected using JSON Web Tokens (JWT) with a defined expiration window. Role-Based Access Control (RBAC) middleware ensures that Voter-only and Admin-only routes are strictly segregated at the API layer.

B. Smart Contract Design

The core security guarantee of the system is implemented through a Solidity smart contract, referred to as Voting.sol, deployed on the Ethereum network. The contract maintains a mapping of election identifiers to candidate vote counts and a separate mapping that records whether a given voter token has already voted in a specific election, thereby enforcing the one-vote-per-election rule directly at the protocol level rather than relying solely on application-level checks. The principal functions exposed by the contract are summarized in Table IV.

TABLE IV. KEY SMART CONTRACT FUNCTIONS

Function	Visibility	Description
createElection()	onlyAdmin	Registers a new election with a unique ID and candidate list
castVote()	external	Records a vote for a candidate; reverts if the voter token has already voted
hasVoted()	view	Returns whether a voter token has already voted in a given election
getVoteCount()	view	Returns the current vote count for a candidate
declareResult()	onlyAdmin	Finalizes and locks the election, emitting a ResultDeclared event

Because the contract enforces the duplicate-vote check using the voter ' s hashed token as part of its own internal state, even a compromised or maliciously modified backend cannot succeed in submitting a second vote on behalf of the same voter, since the blockchain network would independently reject the transaction. This represents the principal security advantage of the proposed architecture over a purely database-driven e-voting system, where the duplicate-vote check exists only in application code and a direct database write could, in principle, bypass it.

C. DataBase Schema (Off-Chan)

While the blockchain stores the authoritative vote record, MongoDB stores all operational and administrative data required to run the platform efficiently. The four primary collections are Users, Elections, Candidates, and Votes (the latter storing only a reference to the on-chain transaction, not the vote choice itself, preserving an additional layer of separation between identity and ballot).

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, June 2026
Impact Factor: 8.2

TABLE V. USERS COLLECTION SCHEMA

Field	Type	Description
_id	ObjectId	Primary key
name	String	Full legal name of voter/admin
email	String	Unique, used for OTP and login
passwordHash	String	Bcrypt-hashed password
role	String	Voter or Admin
age	Number	Validated for 18+ eligibility
aadharDocument	String (path)	Uploaded ID proof (JPG/PNG/PDF)
isVerified	Boolean	Admin verification status
walletAddress	String	Linked Ethereum address for vote signing
createdAt	Date	Registration timestamp

TABLE VII. CANDIDATES COLLECTION SCHEMA

Field	Type	Description
_id	ObjectId	Primary key
name	String	Candidate name
party	String	Party affiliation
electionId	ObjectId (ref)	Linked election
candidateIndex	Number	Index used in the smart contract mapping

TABLE VI. ELECTIONS COLLECTION SCHEMA

Field	Type	Description
_id	ObjectId	Primary key
title	String	Election name
startDate / endDate	Date	Voting window
status	String	Upcoming / Ongoing / Completed
contractElectionId	Number	Reference ID on the smart contract
resultDeclared	Boolean	Whether results have been published

DOI: 10.48175/568

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, June 2026
Impact Factor: 8.2

TABLE VIII. VOTES COLLECTION SCHEMA (OFF-CHAIN REFERENCE)

Field	Type	Description
_id	ObjectId	Primary key
voterTokenHash	String	Anonymized hash of voter identity
electionId	ObjectId (ref)	Linked election
candidateIndex	Number	Selected candidate
txHash	String	Ethereum transaction hash (on-chain reference)
blockNumber	Number	Block in which the vote was mined
timestamp	Date	Vote-cast time

D. Deployment Environment

The completed system was deployed using a fully cloud-based pipeline to validate real-world operability. The React frontend was built using Vite and deployed on Vercel, with the production API endpoint configured through an environment variable. The Express backend was deployed on Render ' s free tier, with environment variables securing the MongoDB connection string, JWT secret, and SendGrid API key. MongoDB Atlas was used as the managed off-chain database, and the Ethereum smart contract was deployed to a test network prior to mainnet consideration, allowing the full request lifecycle — from user action in the browser to blockchain confirmation — to be tested end-to-end.

RESULT

Demo Access:-

Roll	Email	Password
Voter	Dm143dilip@gmail.com	Dilip@123456
Admin	mauryadilip@gmail.com	Kumar@123456

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, June 2026
Impact Factor: 8.2

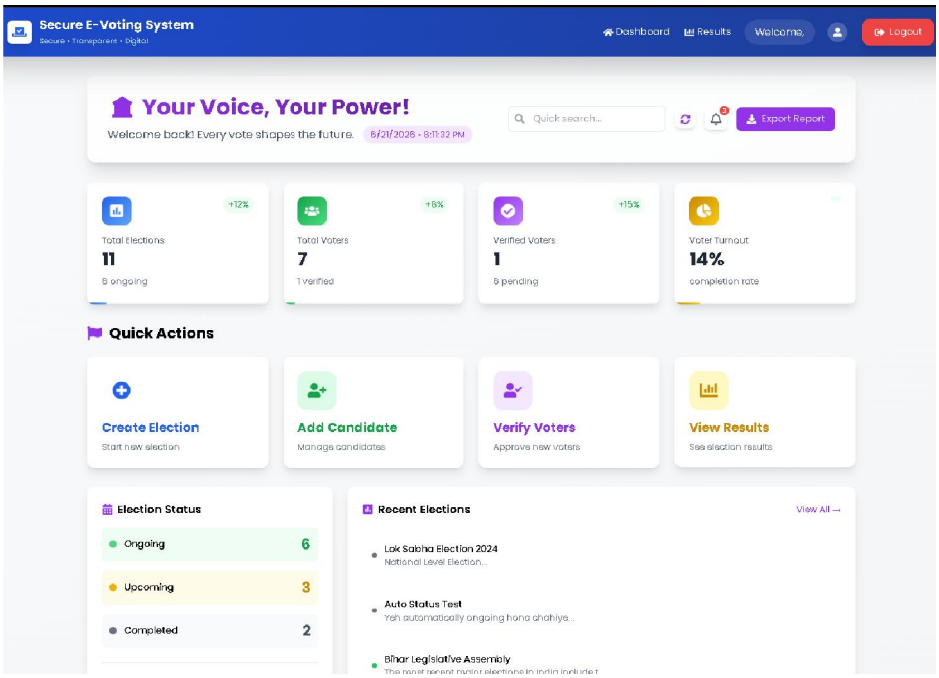

Fig. 1 Admin Dashboard

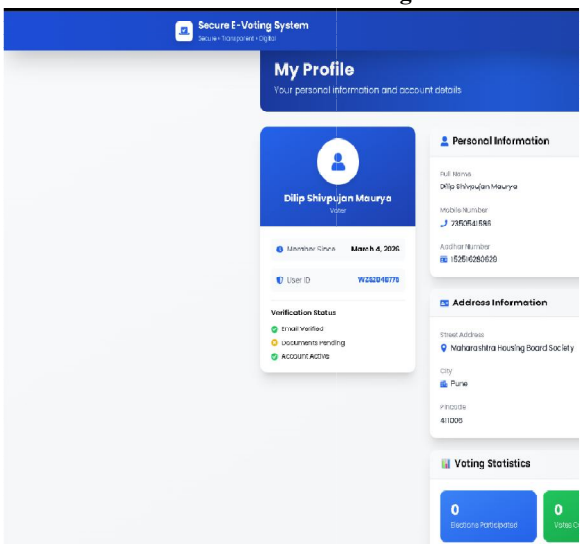

Fig. 2 Voter Profile

API DESIGN AND ENDPOINTS

The backend exposes a RESTful API comprising more than twenty-five endpoints grouped into five functional modules: Authentication, Admin, Voter, Results, and a blockchain-facing Smart Contract Bridge module that translates HTTP requests into Web3 transactions. Table IX summarizes the principal endpoints, their HTTP methods, and the authorization level required for access. All Admin and Voter routes are protected by JWT middleware that validates the bearer token and checks the associated role before granting access to the underlying controller logic.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, June 2026
Impact Factor: 8.2

TABLE IX. SUMMARY OF CORE REST API ENDPOINTS

Module	Endpoint	Method	Auth
Auth	/api/auth/register	POST	Public
Auth	/api/auth/login	POST	Public
Auth	/api/auth/me	GET	Voter / Admin
Auth	/api/auth/verify-otp	POST	Public
Admin	/api/admin/elections	GET, POST	Admin
Admin	/api/admin/elections/:id	PUT, DELETE	Admin
Admin	/api/admin/candidates	GET, POST	Admin
Admin	/api/admin/candidates/:id	PUT, DELETE	Admin
Admin	/api/admin/voters	GET	Admin
Admin	/api/admin/voters/verify	PUT	Admin
Voter	/api/voter/elections	GET	Voter
Voter	/api/voter/vote	POST	Voter
Voter	/api/voter/history	GET	Voter
Results	/api/results/:id	GET	Public
Results	/api/results/:id/declare	PUT	Admin

In addition to these conventional REST endpoints, the Voter Vote endpoint (/api/voter/vote) differs from a typical CRUD operation in that it does not directly write the vote choice to MongoDB. Instead, the controller (i) verifies voter eligibility and election status, (ii) constructs a transaction payload containing the anonymized voter token and candidate index, (iii) signs and forwards the transaction to the deployed Voting smart contract through Web3.js, and (iv) waits for transaction confirmation before returning a success response to the client, at which point only the resulting transaction hash and block number are persisted off-chain for fast lookup. This design ensures that the single most security-critical write operation in the entire system is mediated by the blockchain ratherthan by the application database alone. The complete Postman collection used for endpoint-level testing covers positive-path validation (correct credentials, valid tokens, eligible voters), negative-path validation (expired tokens, duplicate votes, unverified voters attempting to vote), and edge-case validation (malformed Aadhar uploads, election windows that have not yet opened or have already closed), achieving over 95% endpoint coverage during testing.

IV. RESULTS AND DISCUSSION

The proposed system was implemented as a fully functional prototype and evaluated across functional, security, and performance dimensions. This section presents the key observations from testing the deployed application.

A. Functional Testing Results

Functional testing confirmed that voter registration, Aadhar document upload, OTP-based email verification, election creation, candidate management, vote casting, and result declaration all operated correctly across repeated test cycles. Of particular importance, every attempt to cast a second vote within the same election — whether triggered through the user interface or simulated via direct API calls bypassing the frontend — was correctly rejected, confirming that the duplicate-vote protection enforced at the smart contract level functions independently of the application layer

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, June 2026
Impact Factor: 8.2

B. Performance Evaluation

Table X summarizes the key performance indicators recorded during testing. As expected, off-chain operations such as browsing elections or viewing candidate lists completed in well under a second, since they are served directly from MongoDB Atlas. Vote transactions, which must be mined and confirmed on the Ethereum network, naturally required a longer round-trip time; however, this latency remains well within an acceptable range for an election setting, where vote casting is not a high-frequency, time-critical operation in the way that, for instance, a financial trading transaction would be.

TABLE X. PERFORMANCE EVALUATION SUMMARY

Metric	Observed Value	Notes
Average API response time (off-chain)	180 ms	MongoDB-backed CRUD endpoints
Average vote transaction confirmation time	4.8 s	Ethereum test-network block confirmation
Duplicate vote attempts blocked	100%	Enforced at smart contract level
OTP delivery success rate	98.6%	Via SendGrid email service
Aadhar document upload success rate	99.1%	JPG / PNG / PDF, up to defined size limit
Overall endpoint test coverage	95%+	Across 25+ Postman test cases

C. API Test Coverage

Figure 2 presents the proportion of test cases passing per module during Postman-based endpoint testing, demonstrating consistently high coverage across all functional areas, including the blockchain bridge module responsible for translating REST calls into smart contract transactions.

Figure 2 — API Test-Case Pass Rate by Module

Fig. 2. API test-case pass rate by module (25+ endpoints tested via Postman).

D. Security Observations

Three security properties were specifically validated during testing. First, immutability: once a vote transaction was mined, no application-level operation was capable of altering the recorded candidate index, since this would require recomputing the cryptographic hash of every subsequent block — a computationally infeasible task on a properly decentralized network. Second, duplicate-vote prevention: the smart contract' s internal mapping of voter tokens to voting status meant that even a forged or replayed request was rejected at the contract level. Third, credential security: bcrypt hashing ensured that even in the event of a database compromise, raw passwords would not be exposed, while JWT expiration limited the window of exposure for any stolen token

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, June 2026
Impact Factor: 8.2

VII. COMPARATIVE ANALYSIS

To contextualize the contribution of the blockchain security layer, Table XI compares the proposed Blockchain Based Secure E-Voting System against a conventional, purely database-driven e-voting system across eight evaluation criteria.

TABLE XI. TRADITIONAL E-VOTING VS. PROPOSED BLOCKCHAIN-BASED E-VOTING SYSTEM

Criterion	Traditional E-Voting (DB only)	Proposed Blockchain-Based System
Vote Immutability	Dependent on database access controls; an insider with DB privileges can alter records	Cryptographically enforced; altering a vote requires recomputing all subsequent blocks
Duplicate Vote Prevention	Enforced only in application code / DB constraints	Enforced redundantly at the smart contract level, independent of the application layer
Transparency / Auditability	Limited to administrators with database access	Any voter can verify their transaction hash on the public/permissioned ledger
Single Point of Failure	Central database is a single point of failure	Distributed ledger removes single point of failure for vote storage
Performance (routine reads)	Fast (direct DB queries)	Equally fast; routine reads still served from MongoDB
Performance (vote casting)	Very fast (single DB write)	Slightly slower due to blockchain confirmation time
Implementation Complexity	Lower; standard CRUD operations	Higher; requires smart contract development and Web3 integration
Voter Identity Privacy	Voter identity and vote choice may be co-located in the same record	Voter token is anonymized before being linked to a vote on-chain

The comparison highlights a clear trade-off: the proposed system accepts a modest increase in implementation complexity and a few seconds of additional latency at the moment of vote casting, in exchange for substantially stronger guarantees around immutability, duplicate-vote prevention, and auditability. For an application domain such as elections, where the integrity and trustworthiness of the final result is paramount and where vote casting is not a high-frequency operation, this trade-off strongly favors the blockchain-augmented design

V. CONCLUSION AND FUTURE SCOPE

This paper presented the design, implementation, and evaluation of a Blockchain Based Secure E-Voting System that integrates a MERN-stack web application with an Ethereum-based blockchain security layer. By restricting blockchain usage to the vote-casting transaction itself, the system achieves the immutability, transparency, and duplicate-vote prevention benefits of distributed ledger technology without incurring the full performance overhead of running an entire application on-chain. The accompanying authentication pipeline — combining Aadhar document verification, email OTP confirmation, bcrypt password hashing, and JWT-secured sessions — addresses voter authenticity, complementing the vote-integrity guarantees provided by the smart contract layer. End-to-end testing across more than twenty-five REST API endpoints, along with functional and security validation of the deployed prototype, confirmed that the system reliably prevents duplicate voting, maintains high API availability, and produces a verifiable, tamper-evident vote record.

Several directions remain open for future enhancement. First, the current implementation anchors votes to an Ethereum test network; a production deployment would benefit from a detailed gas-cost optimization study and a comparison between public Ethereum, a permissioned consortium chain, and Layer-2 scaling solutions, to identify the most cost-effective option for large-scale elections. Second, the voter-identity-to-wallet linkage could be further strengthened using zero-knowledge proof techniques, allowing a voter to prove eligibility without revealing their wallet address even to the application backend. Third, integrating biometric verification (such as fingerprint or facial

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, June 2026
Impact Factor: 8.2

recognition) alongside Aadhar document checks could further reduce the risk of impersonation during registration. Finally, conducting a large-scale, multi-region pilot deployment would provide valuable real-world data on system behavior under high concurrent load, which was outside the scope of the present prototype-level evaluation.

ACKNOWLEDGMENT

The authors wish to express their sincere gratitude to the Department of Computer Engineering, Alard College of Engineering and Management, Pune, for providing the infrastructure, guidance, and academic support necessary to design, develop, and evaluate the proposed Blockchain Based Secure E-Voting System. The authors also thank the faculty members who reviewed the project at various stages and provided valuable feedback that helped refine both the system architecture and this manuscript.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," white paper, 2008.
[2] V. Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," Ethereum Foundation white paper, 2014.
[3] K. M. Khan, J. Arshad, and M. M. Khan, "Secure digital voting system based on blockchain technology," International Journal of Electronic Government Research, vol. 14, no. 1, pp. 53–62, 2018.
[4] F. P. Hjalmarsson, G. K. Hreioarsson, M. Hamdaqa, and G. Hjalmtysson, "Blockchain-based e-voting system," in Proc. IEEE 11th Int. Conf. on Cloud Computing (CLOUD), 2018, pp. 983–986.
[5] R. Hanifatunnisa and B. Rahardjo, "Blockchain based e-voting recording system design," in Proc. 11th Int. Conf. on Telecommunication Systems, Services, and Applications (TSSA), 2017, pp. 1–6.
[6] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," IEEE Electron Device Lett., vol. 20, pp. 569–571, Nov. 1999.
[7] A. B. Ayed, "A conceptual secure blockchain-based electronic voting system," International Journal of Network Security & Its Applications, vol. 9, no. 3, pp. 1–15, 2017.
[8] N. Kshetri and J. Voas, "Blockchain-enabled e-voting," IEEE Software, vol. 35, no. 4, pp. 95–99, 2018.
[9] M. Pawlak, A. Poniszewska-Maranda, and N. Kryvinska, "Towards the blockchain-based technology in e-voting systems," Procedia Computer Science, vol. 176, pp. 3290–3299, 2020.
[10] M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in Proc. ECOC '00, 2000, paper 11.3.4, p. 109.
[11] OWASP Foundation, OWASP Top Ten Web Application Security Risks, OWASP Foundation, 2021.
[12] D. Maurya, U. Karpe, N. Patil, K. Patil, and J. Mankar, "Secure E-Voting System: A Full-Stack Implementation," project documentation, Alard College of Engineering and Management, Pune, 2026.