# IJEAST

INTERNATIONAL JOURNAL
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY

VOLUME : 10 ISSUE : 10 Print / Issue Publication Date: 08-Apr-2026

ISSN : 2455-2143

DOI : 10.33564/IJEAST.2026.v10i10.009

Indexed In

Google SCHOLAR OPEN ACCESS doi Crossref

WWW.IJEAST.COM

editor@ijeast.com

*International Journal of Engineering Applied Sciences and Technology, 2026*
**Vol. 10, Issue 10, ISSN No. 2455-2143, Pages 56-64**
**Published Online February 2026 in IJEAST (http://www.ijeast.com)**

# DECENTRALIZED VOTING SYSTEM USING BLOCKCHAIN

Danish Ahmed, Taha Iftikhar, Mohd Farhan Khan, Zayan Hashmi, Yahiya Bakhtiyar Ahmed Siddiqui
Department of Computer Science & Engineering Integral University
Lucknow, India

*Abstract*— **Student representative elections are a critical form of campus governance in colleges, yet many traditional election voting methods often contain many issues such as being lacking in transparency, hard to access, and mistrusted. This research paper explores and implements a blockchain-based voting system that is designed for use in college and university elections. The proposed voting system has been developed using a Node.js backend, a React-based front end, and Ethereum smart contracts to create an auditable and secure electronic ledger of vote cast. Importantly, students will be able to use existing institutional databases (e.g., student record databases and enrollment databases) for identity verification, which means that there does not need to be any reliance on external identity verification vendors nor an excessive amount of centralized administration associated with identity verification. This will allow eligible students registering to vote using existing data to their institution while maintaining a decentralized nature of the actual voting process. We describe in detail the architecture of the voting system, the functionality of the smart contract logic, both the emergency control functions as well as the vote-casting functions, the structure of the database schema associated with managing off-chain entities, and the protocols needed to integrate with the institution's databases. Similarly, we present a security evaluation of the blockchain- voting solution we have implemented to highlight how utilizing existing institutional information reduces the amount of potential single points of failure when compared to traditional identity verification methods which rely on administrative staff for validation. Furthermore, the results indicate that there is a high degree of auditability and an unparalleled degree of accessibility for campus elections via this hybrid voting mechanism, while also providing a high degree of assurance of voter anonymity.**

*Keywords*-- **electronic voting, blockchain, Ethereum, smart contracts, decentralized applications, student elections, institutional databases, campus governance**

## I. INTRODUCTION

Student elections are an essential component of democratic process on college campuses, providing an opportunity for students to elect representatives to serve on student councils, in clubs, and in governing bodies. Unfortunately, traditional paper-based voting processes as well as basic electronic voting systems do not offer reliable methods of assuring transparency; thus, making it very difficult for students to verify the accuracy of results and ultimately trust the outcome of the election process. The application of blockchain technology serves as a unique and compelling solution by offering multiple levels of transparency, immutability and distributed ledger for recording ballots. This project proposes an integrated and end-to-end voting system specifically for academic institutions (colleges and universities). By creating an ecosystem that offers the full range of benefits derived from utilizing blockchain technology, while at the same time incorporating existing institutional databases for verification of identity, allows for a full range of benefits to be derived from both systems. Smart contracts will be utilized to record elections, candidates and votes on an Ethereum-compatible distributed ledger. A Node.js-based back-end will facilitate the interaction between clients and the blockchain and an identity layer will leverage the institution's student database (i.e. enrollment records and student ID's) to verify the identities of eligible voters without utilizing external identity providers. The GitHub repository will contain all the necessary Solidity contracts for the voting logic; deployment and compilation scripts; API containing middleware for vali- dating and authenticating the institutional data; database models presenting both voters and elections; as well as a separate front-end workspace to provide voter and candidate interfaces. The current paper outlines the overall system architecture, as well as providing justification for combining on-chain voting with institutional verification of data, and describes the major contributions made to this area. It also includes an audit of the decisions made during the development of the project, presents the evaluation methodology that was used to evaluate the goals of the project, and offers future directions for study.

### A. **Background and Motivation**

*International Journal of Engineering Applied Sciences and Technology, 2026*
**Vol. 10, Issue 10, ISSN No. 2455-2143, Pages 56-64**
**Published Online February 2026 in IJEAST (http://www.ijeast.com)**

The challenges faced by academic institutions when it comes to student elections are substantial, including limited time windows to vote; widely dispersed populations of students across campuses; and the need to provide timely and transparent results. Traditional means of voting are often limited by the fact that they may not provide easy access to the voting process (e.g., since students may not be on campus during the voting period) and they often create problems of trust (e.g., the accountability of manual counting means that it may be difficult to verify the counted votes). Both blockchain technology and institutional data provide an innovative approach to resolving these issues. Most, if not all, academic institutions already maintain large volumes of records about their students. This includes: student enrollment status in each program that the student is a part of; their student ID; and their eligibility to vote. By utilizing this existing infrastructure of records, this system does not need external identity providers - removing any dependence on centralized administrative approval that would have created a bottleneck or a single point of failure.
That is the motivation for this project - to develop a voting system that meets the requirements of being both transparent (due to the use of a blockchain-based voting system) and a practical solution by utilizing readily available institutional data.

### B. **Problem Statement**

Ensuring secure and auditable student elections involves the conflicting requirements of voter privacy versus public verifiability and accessibility versus Sybil resistance (the ability to limit one person from voting several times). Traditional admin-centric verification systems can introduce centralization risk since if an election administrator's account is hacked or biased, the entire integrity of the election is jeopardized. Therefore, the main research question for this work is: What is the best way to design a voting system that will use institutional databases to verify eligible voters while avoiding centralization and maintaining the secrecy of voted ballots? By integrating with existing college databases, the proposed design can automatically verify a student's enrollment status without requiring manual administrator approval for each in-dividual voter, thereby distributing trust and eliminating single points of failure within the election process.

## II. LITERATURE REVIEW

### A. **Electronic Voting Systems**

Electronic voting systems have been proposed as an option for enhancing the accessibility, efficiency, and auditability of elections. Chaum [1] pioneered the idea of untraceable electronic mail and digital pseudonyms, which was thus established the groundwork for anonymous or privacy-preserving electronic voting. The assessment of electronic voting systems and programs [6] shows a variety of methods to strike a balance between security, privacy, and usability in electoral settings. The latest verification methods for electronic voting, including Helios [8] and Belenios [9], have proven that it is possible to hold an open-audit, end-to-end verifiable election [7] while still respecting voters' privacy.

### B. **Blockchain-Based Voting**

Several proposals for blockchain-based voting systems have been created that will allow for decentralized ballot casting [10] and tamper-proof [12] counting by utilising distributed ledger technologies. [11] The advent of blockchain technologies have revived the interest in decentralised voting systems. McCorry et al. [2] introduced a smart contract for boardroom voting with maximum privacy assurance for voters thus proving that voting logic is able to be implemented on blockchain based platforms. The work by Kiayias et al. [4] presented Ouroboros, a provably secure proof of stake blockchain protocol which guarantees security properties through formal proofs. These efforts indicate that blockchain technologies will be able to allow for transparent, tamper-evident records made available to the public while maintaining the privacy of voters through the use of cryptographic techniques.

### C. **Privacy-Preserving Techniques**

Privacy-preserving techniques are important methods to ensure the anonymity of voters within an electronic voting system. Zyskind et al. [3] investigated how blockchain could protect personal data by decentralizing it. Paillier [5] described the development of public-key cryptosystems that are based on composite degree residuosity classes, enabling the process of homomorphic encryption and potentially useful in developing privacy-preserving electronic voting systems. Additional cryptographic techniques, such as receipt-free [13] voting protocols, anonymous credential systems [14], and zero-knowledge proof constructions [15], provide additional security for secret voting in decentralized systems. These techniques allow for the separation of voter identity from the content of their ballot while still ensuring the authenticity of the vote through verifiable methods.

## III. METHODOLOGY

### A. **Repository Overview**

The repository that was used for this analysis consists of a blockchain-based voting system containing a total of three components: smart contracts written in Solidity, a Node.js backend and, a frontend developed using Vite and React. Smart contracts define all major aspects of the voting system's logic, such as candidate registration, voter registration, and vote casting. The backend contains RESTful API endpoints used to perform election-related operations and also includes middleware to validate identity via the Aadhaar scheme. The frontend provides separate interfaces for voters and candidates to interact with the voting system.

*International Journal of Engineering Applied Sciences and Technology, 2026*
**Vol. 10, Issue 10, ISSN No. 2455-2143, Pages 56-64**
**Published Online February 2026 in IJEAST (http://www.ijeast.com)**

B. **Code Review Approach**
The code review comprised a combination of three primary focus areas: 1) Security and correctness of the smart contracts; 2) Architecture of the backend and implementation of middleware; and 3) User experience and design of the frontend. With respect to the smart contracts, consideration was given to examining access control pattern use, the execution of state transitions, and proper validation of input. For the backend, consideration was given to assessing the efficacy and appropriateness of the API routes, authentication and authorization mechanisms, and database model design. And, finally, for the frontend, consideration was placed on evaluating the component architecture, state management and user interaction flows.

C. **Analysis Methods**
This analytical approach combined structured code review with selective execution of scripts provided in the repository. The code review focused on three areas: the interface boundaries, the data flows between the identity services and the blockchain, and the explicit storage of identifying attributes within the code. Examples of the types of analyses performed on the smart contract codes were accessing control patterns and state transition logics, as well as common pitfalls associated with the Solidity programming language. The backend code was reviewed for input validation, access authentication, and use of middleware layers. Once the scripts contained in the repository were confirmed as being runnable in a local environment without any external dependencies, they would be executed to verify the expected artifacts (e.g., compiled contract JSON files and deployment metadata) were produced. The analysis also compared the practices included in the repository against best practices found across literature.

D. **Validation Approach**
To validate the results of the analysis, the researcher cross-checked all static findings by verifying all executable artifacts and to compare the behavior of the repository against the documented expectations of the code and scripts. For example, the presence of compiled contract artifacts and deployment JSON files would provide evidence that the compilation and deployment scripts were run in the development history. The study intentionally refrained from conducting tests that would engage external identity services, thereby confining dynamic validation to local-only flows to prevent interaction with production systems. The reliability of the findings was enhanced by obtaining corroboration from multiple files, such as confirming that routes invoking Aadhaar validation aligned with middleware implementations and model updates.

E. **Automated Testing Strategy**
The repository includes a comprehensive testing suite located in backend/scripts, notably test-simplified-voting.js. This script utilizes the Ethers.js library to interact with a local Ganache blockchain, simulating a complete election lifecycle. The automated tests cover 11 distinct scenarios:

**1) Registration:** Verifies that candidates and voters can register distinct identities and checks preventing duplicate registrations.
**2) State Transitions:** Validates that the election state transitions correctly from “Not Started” to “Active” to “Ended”.
**3) Voting Logic:** Simulates multiple voters casting ballots for different candidates and ensures the vote counts update atomically.
**4) Privacy Checks:** Confirms that a voter cannot vote twice and that their choice is recorded against their address (internal view only).
**5) Results Announcement:** triggers the tallying algorithm to identify the winner with the most votes. Emergency Procedures: A crucial test case launches an election, deploys a new contract, initiates a emergencyStop to stop voting, and then confirms that no more votes can be cast until resumeVoting is called.
This script offers a high level of assurance regarding the smart contract’s resilience against state-machine violations and functional correctness.

## IV. SYSTEM ARCHITECTURE

A Presentation Layer (Frontend), an Application Layer (Backend Identity Service), and a Data Persistence Layer (Blockchain Database) make up the three-tier architecture of the suggested system.

A. **Frontend Layer**
For high-performance builds, the user interface is constructed with **React.js** in conjunction with **Vite**. **Tailwind CSS** is used to manage styling, guaranteeing a responsive design on all devices. Three separate portals comprise the frontend:

- **Voter Portal:** Allows users to register, connect their Web3 wallet (e.g., MetaMask), view active elections, and cast votes.
- **Candidate Portal:** Enables candidates to register, create profiles, and track their vote counts.
- **Admin Dashboard:** Provides system administrators with tools to deploy new election contracts, verify voter documents, and manage emergency controls.

Authentication on the client side is handled via a dual mechanism: standard JWT tokens for session management and Web3 wallet signatures for blockchain interactions.

*International Journal of Engineering Applied Sciences and Technology, 2026*
**Vol. 10, Issue 10, ISSN No. 2455-2143, Pages 56-64**
**Published Online February 2026 in IJEAST (http://www.ijeast.com)**

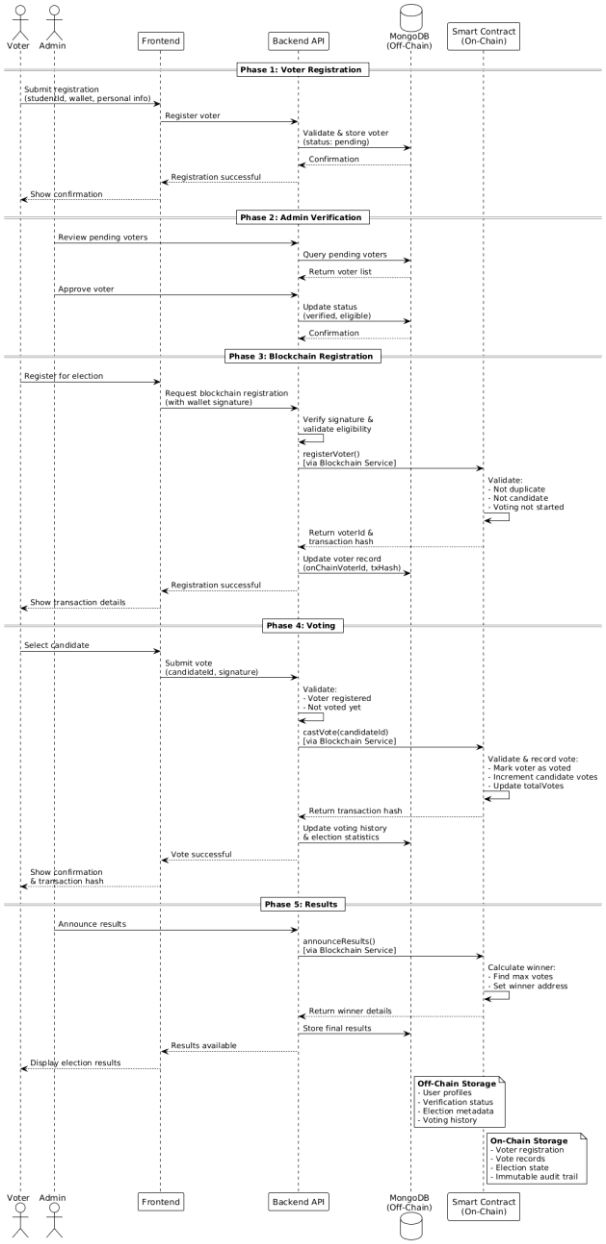


Fig. 1. Voting Flow Diagram

B. **Backend Application Layer**

Using **Node.js** with the **Express.js** framework, the backend connects to the off-chain world (i.e. user identity and metadata) and to the on-chain world (i.e. smart contracts). Some responsibilities of the backend are:

1. **API Services:** Provide RESTful API endpoints so that users can register, the elections can be managed and users can retrieve their election results.
2. **Middleware Integration:** Provide Middleware Integration in the form of custom middleware to provide Aadhaar-based identity verification ensuring that only valid citizens can register.
3. **Smart Contract Interaction:** Provide libraries for smart contract interaction (likely Ethers.js or Web3.js). These libraries allow the backend to listen to smart contract events and keep the off-chain database in sync with the on-chain contract state.

C. **Blockchain Layer**

Voting on an Ethereum [16] Blockchain employs voting logic as a programmable smart contract [17] to execute de-centralized applications. The voting logic will exist on an Ethereum-compatible blockchain. For development and testing, the project will use either **Ganache** or **Anvil** to create a local blockchain for simulation purposes. The project's primary smart contract is called SimplifiedVotingSystem.sol and will manage the state of elections, candidates and votes. Previous examples of voting with blockchain technology have demonstrated that candidate registration, vote casting, and the automated tallying of votes can all be performed through smart contract transactions in the blockchain [19].

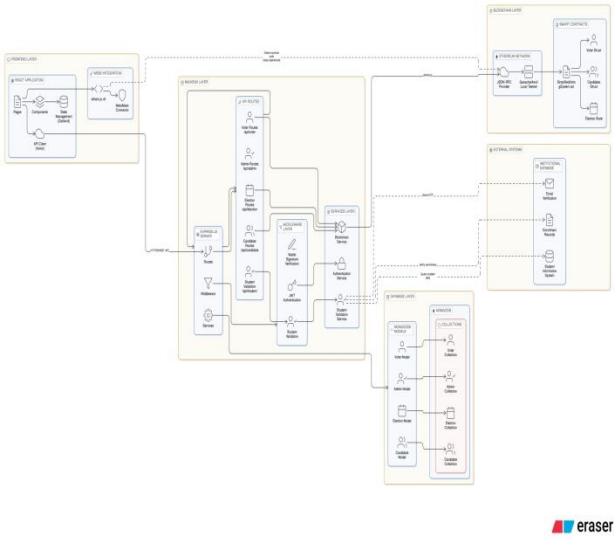


Fig. 2. High-Level System Architecture showing the flow from Client to Backend/Identity Service and finally to the Blockchain.

## V. IMPLEMENTATION DETAILS

This section provides the specifics of the technical implementation of the core components of the system.

A. **Smart Contract Logic**

The SimplifiedVotingSystem.sol contract is the core of the system. It contains the state variables and functions that will operate securely.

*International Journal of Engineering Applied Sciences and Technology, 2026*
**Vol. 10, Issue 10, ISSN No. 2455-2143, Pages 56-64**
**Published Online February 2026 in IJEAST (http://www.ijeast.com)**

1. Data Structures: The contract uses "Structs" to establish key entities of the system:

```
[language=solidity]
struct Voter {
uint voterId;
uint voteCandidateId; address voterAddress; bool isRegistered; bool hasVoted;
}
struct Candidate {
uint candidateId;
address candidateAddress;
uint votes;
bool isActive;
}
```

Throughout the contract, mappings allow access to these structs quickly using O(1) lookups by ID and address.

2. Key Functions:

- registerVoter(): The function allows you to regis- ter an address, it checks the sender has not already been registered by using the voterAddressToId mapping so double registrations will not happen.
- castVote(uint _candidateId): This function allows the vote to be executed. Importantly it also checks:
- The voter is registered.
- The voter has not already voted.
- The election is currently active.
- The candidate ID is valid.

After a successful vote cast, the candidate's vote tally is updated. A VoteCast event will be triggered at this stage.

- emergencyStopVoting(): Restricted access to the contract at the electionCommission address. Al- lowing the administrator to disable the contract when a vulnerability or attack is detected.

B. **Off-Chain Database Schema**

For the sake of storing data that is too costly to keep on-chain or that may not be suitable to keep on-chain (e.g.) database storage of student data; Names, Student IDs, Enrollment) the system has been designed using **MongoDB**.

1) Voter Model: The Voter.js contains the following fields that will reference the voter.

- studentId: The ID of the student from their previous institution (enrollment number) and will be matched against the database maintained by the institution to validate the student against their college records.
- walletAddress: The Ethereum address associated with the student.
- votingHistory: Will contain an array of transaction hashes that connect the voter to the voter's actions on-chain.
- email: The e-mail associated with the student and used to validate and notify the student about the voting process.

**Note:** Student IDs will have a schema reference with existing records maintained by the educational facility, which would allow for a lower reliance on centralized administration to verify enrollment status.

C. **Identity Verification System**

The identity layer exists as a gatekeeper to access the institution's existing student database. The identity layer will be found in the validation middleware.

1) **Institutional Database Integration:** he college's stu- dent information system (SIS) connects to the voting system through either the SIS or through a simulated data set containing student IDs, enrollment status, and email addresses. This eliminates any external identity providers.

2) **Validation Logic:** When students register, the system validates:

- The format of their student ID (as determined by the institution).
- Their enrollment status by querying an institutional database.
- Whether they have previously registered.
- Sends a verification email to their institutional email address.

3) **Decentralized Trust:** By utilizing existing institutional data, the voting system has a reduction in reliance on a single admin account. The verification of voters takes place against the database and is automatic and less prone to bias by an administrator or a single point- of-failure that could allow an administrator to block qualified voters from registering.

4) **Linkage:** Once a student's ID is validated, it is linked to the student's wallet address in the MongoDB "Voter" collection to ensure one person cannot register multiple wallets.

D. **Frontend Implementation**

The user interface (UI) is an essential component for overall system adoption and is implemented using **React.js** (v18) and **Vite** for optimized bundling.

1) Component Architecture: The user interface codebase follows a modular architecture.

- **State Management:** Zustand is being used for global state management without the boilerplate associated with Redux, lightweight handling of user session states (ad- min, voter, candidate) and wallet connection states.
- **Routing:** React Router v6 handles navigation be- tween the public landing page, protected voter registration routes, and the secure admin dashboard. The high-level

layout wrapper components are responsible for providing consistent footer and navigation bars according to the active user role.

- **Web3 Integration:** ethers.js (v6) is used to interface with the Ethereum network, and custom hooks manage the entire connection lifecycle to facilitate user-connected network switching when not connected to the correct chain (e.g. local Ganache instance).
- **UI/UX:** Tailwind CSS assists with utility first styling to ensure responsive design while providing a set of consistent icons through lucide-react and using Sonner to manage toast notifications for immediate user feedback on transaction status (pending, success, failed).
- **Form Handling:** React Hook Form (with Zod schemas) manages complex registration forms by providing real-time client-side validation for Aadhaar number fields, name fields, and age fields, and preventing these forms from being submitted to the backend until they are valid.

## VI. SECURITY ANALYSIS

The hybrid architecture imposes certain security properties and risks.

### A. **Immutability and Transparency**

BWith votes being cast on the Ethereum blockchain, they are immutable once they have been recorded, preventing alterations or deletions after they have been sent to the blockchain. The Events that are triggered when a vote is cast create a trail that can be easily verified against the final tally by anyone with access to the Ethereum blockchain; therefore, the process is fully auditable by any member of the public.

### B. **Reduced Centralization Through Institutional Data**

Using the existing institution's student database for verification means that the system has reduced risks associated with centralization compared to models that are dependent on administrators. When there is only one admin (or group of admins) able to create voter records and check eligibility (i.e., verify the identity of each voter) in a traditional voting process, there is a considerable risk of there being insufficient or abusive processes. By using the institutional student database, verification will occur automatically through the institutional infrastructure that is currently in place, thus distributing the trust of the verification process. Throughout the contract there will remain an 'onlyCommissioner' modifier for functions, such as 'startVoting' and 'announcingResults', but the creation of a voter record and their eligibility will not be reliant on the approval of administrators thus eliminating the potential for single points of failure.

### C. **Privacy and Data Protection**

Conceptual frameworks for vote casting and verifying the outcome of elections using Blockchain technology have demonstrated that using distributed ledgers can improve the integrity of elections while keeping the confidentiality of the identity of those voting through cryptographic protections. The protection of student data is a very important consideration in the implementation of the voting application. The use of studentId in the 'Voter.js' model creates a direct relationship between studentId and 'walletAddress'.

- **Risk:** If an attacker gained access to the MongoDB database, they could link an on-chain vote (as identified by walletAddress) to a specific student via their studentId.
- **Mitigation Strategies:**

1. Store only hashes (i.e., "fingerprints") of studentId and not the actual studentId.
2. Implement role-based access control (RBAC) on the database so that individuals can only query voter-wallet mappings based on their roles.
3. Utilize zero-knowledge proofs (ZK-SNARKs) to demonstrate eligibility for enrollment without sharing the specific student ID.
4. Use the institutional database for initial verification, and then provide anonymous voting credential to participants

- **Advantage over External Systems:** Unlike external identity providers (such as national identity providers), the institutional databases are already subject to campus privacy policy (such as FERPA in the USA, or equivalents in other countries) thus establishing a legal and technical basis for safety of the data they contain.

## VII. RESULTS AND DISCUSSION

### A. **Implementation Results**

The repository is an example of an effectively designed hybrid voting system for educational institutions. It integrates deployment scripts, front-end parts, backend parts, and sensible contracts to bring about a deployed prototype. The voter registration, candidate registration, and casting of ballots are all implemented by Solidity sensible contracts. The evidence of deployed sensible contract artifacts in (contracts/compiled) and deployment metadata in (backend/deployments) suggests a deployed automated workflow for contract management. Furthermore, the backend includes REST endpoints for the execution of election operations and middleware that includes institutional database validation. This is a real-world application of gate voter registration through campus data infrastructure. This design choice removes the need to depend on external identity providers and utilizes existing student enrollment information. The frontend code shows good usage of current development tooling and separation

*International Journal of Engineering Applied Sciences and Technology, 2026*
**Vol. 10, Issue 10, ISSN No. 2455-2143, Pages 56-64**
**Published Online February 2026 in IJEAST (http://www.ijeast.com)**

of interface responsibilities between voters, candidates, and administrators.
Static analysis of the smart contract code revealed standard input validation and access-control patterns: the textttonlyCommissioner modifier. The project also included extensive test scripts,
texttttest-simplified-voting.js, which cover 11 different scenarios, from registration to results announcement. However, functional flows—rather than adversarial or fuzz testing—are the main focus of test coverage. Compilation and deployment are managed by multiple scripts, enhancing reproducibility and reducing the risk of manual deployment.

B. **Security Analysis Results**
The implementation shows both areas for improvement and strengths from a security perspective: **Advantages:**

- **Less Centralization:** The system decentralizes bottle-necks and the possibility of administrative bias by dis-tributing trust beyond a single admin account by means of automated verification against institutional databases.
- **Immutable Vote Records:** Strong auditability is pro-vided. by on-chain storage that guarantees votes cannot be altered after they are minted.
- **Governance Existing Framework:** Using databases of institutions, student-level data that is already protected by campus privacy policies (such as FERPA), and through technical and legal protections.

**Identified Risks:**

- **Identity-Address Linkage:** The model Voter.js links studentId directly to walletAddress. If the MongoDB database is compromised, then attackers could correlate on-chain votes with specific students.
- **Inadequate Privacy Controls:** The lack of crypto-commitment schemes of graphical or zero-knowledge proofs increases the risk of deanonymization.
- **Commissioner Centralization:** While the registration is decentralized, critical functions StartVoting, announceResults) remain under single-account control.

In all, the repository provides a working prototype. with solid engineering scaffolding, but would benefit from improve the state-of-art in privacy protection, adversarial testing, and formal pre-production deployment contract audits.

C. **Discussion**
The project is an example of the practical synthesis of institutional theory in action. data validation and decentralized record-keeping, notably adapted for campus elections. End- to-end verifiable voting systems have shown that the application of cryptography as a means of verifying whether an election has occurred accurately does not violate the individual's right to vote anonymously [18]. This hybrid design reduces operational burdens through utilizing existing campus infrastructures for eligibility checks, and utilizing blockchain contracts to provide a transparent vote ledger. The approach is particularly well suited for academic institutions where:

1. The student databases are already maintained and regularly Updated.
2. Elections will have smaller populations (hundreds to thousands instead of millions).
3. Existing systems used to manage IT governance can work with all types of voting systems (systems used for voting).
4. A trust-based administration of institutions provides a stable foundation for hybrid architectures.

The transparent nature of blockchains is at odds with the traditional secrecy offered by paper ballots, and the patterns emerging from current implementations are a testament to this contradiction. The on-chain storage of election and voter registration state creates certain constraints for properly en- cumbering the use of personally identifiable information (PII) or deterministic IDs (IDs that allow an identity match) between off-chain verifications and on-chain vote records.
The functional architecture of the repository is composed of proper separation of concerns among API routes, middleware, models, and smart contract logic, facilitating accurate testing and iterative development. The modularity of the architecture allows the agency to reconfigure the verification layer of the voting systems to their unique student information systems without having to alter the core voting code logic itself.
There are some key gaps that need to be addressed before fully deploying this system:

- **Formal Verification:** Having mathematical proofs of the contract's invariant properties would provide a stronger guarantee against logic flaws than the standard test guarantees commonly used with software development.
- **Adversarial Testing:** Adversarial testing is useful in exposing the edge cases that unit tests do not normally detect.
- **Privacy Enhancements:** Zero-knowledge proofs or anonymous credential systems would eliminate the connection between the voter and the results from the voting records.
- **Governance Decentralization:** The use of either multi-sig wallets or structures as DAOs for the position of the Election Commissioner will result in less centralization.

Lastly, social and regulatory implications must be considered. While Institutional databases have exclusive bene- fits, their incorporation with electronic voting systems must take into consideration student privacy rights, data retention policies, and transparency in governance of the

*International Journal of Engineering Applied Sciences and Technology, 2026*
**Vol. 10, Issue 10, ISSN No. 2455-2143, Pages 56-64**
**Published Online February 2026 in IJEAST (http://www.ijeast.com)**

verification process. The system must have audit trails documenting when/how student data was accessed for verification purposes and the students must have visibility regarding what information they have used and how it is protected.

## VIII. CONCLUSION

A thorough examination of a blockchain-based voting system created especially for educational institutions was provided in this paper. The implementation shows a working prototype that combines Solidity smart contracts with a contemporary MERN stack (MongoDB, Express, React, and Node.js). The system obtains a number of significant benefits by using the institutional databases already in place for student verification, including integration with current campus data governance frameworks, automatic eligibility verification against enrollment records, and a decreased reliance on centralized admin approval. Although the system's hybrid architecture effectively achieves immutability and avoids double-voting, our analysis revealed areas where privacy protection could be strengthened. Future research ought to concentrate on:

1. Using cryptographic hashes in place of plaintext student ID storage.
2. To fully decouple identity from voting records, zero-knowledge proofs are being used for enrollment verification.
3. Using DAO governance structures or multi-signature wallets to decentralize the "Election Commissioner" position.
4. Performing official penetration tests and security audits prior to production deployment.

The system is a sensible attempt to modernize student elections by striking a balance between the advantages of blockchain transparency and the practicalities of campus IT infrastructure. The architecture provides academic institutions with a feasible way to introduce safe, auditable, and easily accessible electronic voting by leveraging their current institutional data systems rather than relying on outside identity providers.

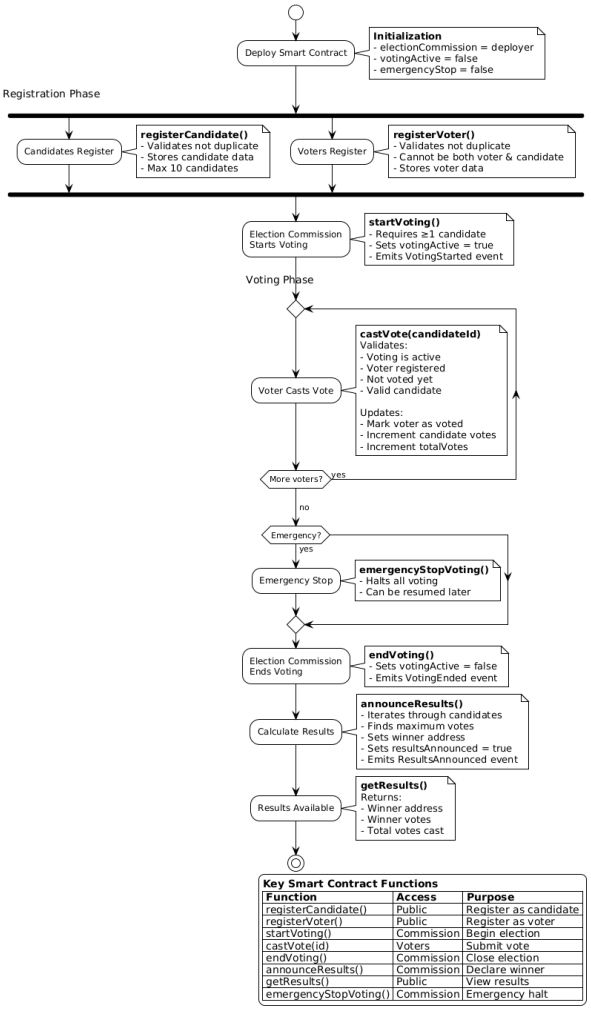


Fig. 3. Smart Contract State Machine

## IX. ACKNOWLEDGMENTS

The authors would like to acknowledge the contributors to the open-source repository that made this analysis possible, and the maintainers who provided documentation and scripts that supported the audit. Input from colleagues in the areas of cryptography and distributed systems helped shape the analysis and recommendations presented here.

## X. REFERENCES

[1]. Chaum D. L., (1981), "Untraceable electronic mail, return addresses, and digital pseudonyms," Communications of the ACM, (vol. 24, no. 2, pp. 84–90).

[2]. McCorry P., Shahandashti S. F., and Hao F., (2017), "A smart contract for board- room voting with maximum voter privacy," in Financial Cryptography and Data Security, Springer, (pp. 357–375.)

[3]. Zyskind G., Nathan O., and Pentland A., (2015) "Decentralizing privacy: Using blockchain to protect personal data," in Proceedings of the IEEE Security and Privacy Workshops, IEEE, (pp. 180–184).

[4]. Kiayias A., Russell A., David B., and Oliynykov R., (2017), "Ouroboros: A provably secure proof-of-stake blockchain protocol," in Annual International Cryptology Conference (CRYPTO), Springer, (pp. 357–388).

[5]. Paillier P., (1999), "Public-key cryptosystems based on composite degree residuosity classes," in Advances in Cryptology — EUROCRYPT, Springer, (pp. 223–238).

[6]. Cabuk U. C., Adiguzel E., and Karaarslan E., (2020), "A survey on feasibility and suitability of blockchain techniques for the e-voting systems," arXiv preprint (arXiv:2002.07175)

[7]. Benaloh J., (2006), "Simple verifiable elections," in Proc. USENIX/ACCURATE Electronic Voting Technology Workshop, (pp 1-10)..

[8]. Adida B., (2008), "Helios: Web-based open-audit voting," in Proc. 17th USENIX Security Symposium, (pp. 335–348).

[9]. Cortier V., Galindo D., Glondu S., and Izabachene M., (2016), "Belenios: A simple private and verifiable electronic voting system," in Proc. Foundations of Security Analysis and Design (FOSAD), (pp 214–238).

[10]. Hardwick F. S., Gioulis A., Akram R. N., and Markantonakis K., (2018), "E- voting with blockchain: An e-vote casting protocol," in Proc. 5th Int. Conf. Internet Technology and Secured Transactions (ICITST), (10.1109/Cybermatics_2018.2018.00262).

[11]. Kshetri N. and Voas J., (2018), "Blockchain-enabled e-voting," IEEE Software, vol. 35, no. 4, (pp. 95–99).

[12]. Ayed A. B., (2017), "A conceptual secure blockchain-based electronic voting system," Int. J. Network Security Its Applications, vol. 9, no. 3, (pp. 01–09).

[13]. Benaloh J. and Tuinstra D., (1994), "Receipt-free secret-ballot elections," in Proc. 26th ACM Symposium on Theory of Computing, (pp. 544–553).

[14]. Camenisch J. and Lysyanskaya A., (2001), "An efficient system for non- transferable anonymous credentials with optional anonymity revocation," in Advances in Cryptology – EUROCRYPT, (pp. 93–118).

[15]. Groth J., (2010), "Short non-interactive zero-knowledge proofs," in Advances in Cryptology – ASIACRYPT, (pp. 341–358).

[16]. Buterin V., (2014), "Ethereum: A next-generation smart contract and decentralized application platform," Ethereum White Paper, (pp. 13-21).

[17]. Wood G., (2014), "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Yellow Paper, (pp. 1-42).

[18]. Kiayias A., Zacharias T., and Zhang B., (2015), "End-to-end verifiable elections in the standard model," Advances in Cryptology – EUROCRYPT 2015, Lecture Notes in Computer Science, vol. 9056, (pp. 468–498).

[19]. M. Hjalmarsson, G. Hreiarsson K., Hamdaqa M., and Hja´lmty´sson G., (2018), "Blockchain-based e-voting system," in Proc. IEEE 11th Int. Conf. Cloud Computing (CLOUD),(pp. 983–986).

# IJEAST

## INTERNATIONAL JOURNAL
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY

### ABOUT IJEAST

International Journal of Engineering Applied Science and Technology (IJEAST) is a peer-reviewed, open access journal that publishes high-quality research papers in the field of Engineering, Applied Science and Technology.

IJEAST aims to provide a platform for researchers, academicians, and professionals to share their innovative ideas, research findings, and practical experiences with the global scientific community.

### FOCUS AREAS

- Engineering
- Applied Science
- Technology
- Innovation & Development
- Interdisciplinary Studies

### PEER REVIEWED
All submissions are rigorously peer reviewed to ensure quality.

### OPEN ACCESS
Free and unrestricted access to research for all.

### GLOBAL REACH
Connecting researchers and professionals worldwide.

### TIMELY PUBLICATION
We ensure a swift and efficient publication process.

For more information, visit our website
**www.ijeast.com**

**INTERNATIONAL JOURNAL**
OF ENGINEERING APPLIED SCIENCE
AND TECHNOLOGY

editor@ijeast.com
www.ijeast.com
India

2455-2143