VoteChain

Manroop Singh, Kevin Dhankhar, Mokksh Kapur, Vaibhav Sheoran
SRM University, Sonipat

Abstract
VoteChain is a lightweight, blockchain-enabled voting system designed for secure and transparent elections in local or controlled network environments. The platform separates administrator and voter functionalities, enabling role-based authentication, candidate management, election lifecycle control, and one-vote-per-voter enforcement. Votes are recorded as immutable blockchain transactions, ensuring auditability and tamper resistance without relying on external distributed networks. A simple web-based interface supports ease of use, while the backend integrates FastAPI, JWT authentication, and a custom blockchain engine. VoteChain demonstrates how decentralized principles can strengthen trust in small-scale digital elections.

Contents

1 Introduction

1.1 Background

Electronic voting has become increasingly important as institutions seek faster, more efficient, and accessible ways to conduct elections. However, many conventional e-voting systems face issues such as data manipulation, lack of transparency, and limited auditability. Blockchain technology, with its decentralized and tamper-proof characteristics, has been proposed as a viable method to enhance trust and security in such systems. VoteChain adopts these principles in a simplified, self-contained form suitable for campus environments, organizations, and controlled network scenarios.

1.2 Problem Statement

Traditional digital voting platforms often rely on central databases where votes are stored in easily alterable formats, making them vulnerable to tampering and unauthorized modifications. Additionally, many systems lack clear role separation, secure authentication, and verifiable audit trails. There is a need for a compact, easy-to-deploy solution that ensures integrity, transparency, one-vote-per-voter enforcement, and clear oversight without requiring external distributed networks or complex infrastructure.

1.3 Objectives

The primary objective of VoteChain is to develop a secure and transparent voting platform using blockchain principles. Specific aims include: enabling role-based access for administrators and voters; ensuring that each registered voter can cast exactly one vote; providing immutable vote storage using block-linked transactions; offering a simple web interface for ease of use; supporting full election lifecycle management; and generating verifiable results with turnout statistics.

1.4 Scope of the Projecr

VoteChain is designed for small-scale, local-network elections such as classroom, departmental, or organizational polls. The system focuses on clarity and security rather than large-scale distributed blockchain deployment. Its scope includes authentication, voter registration, candidate management, blockchain-backed vote recording, election state control, and result generation. External node synchronization, nationwide scalability, advanced consensus, and real-world political election use are outside the scope of this project.

---

2 System Overview

2.1 Concept of Electronic Voting

Electronic voting refers to the use of digital technologies to cast, record, transmit, and count votes electronically instead of relying on traditional paper-based methods. E-voting systems aim to improve efficiency, reduce human error, and offer rapid result computation. A typical electronic voting workflow includes voter authentication, ballot presentation, vote submission, secure storage, and final tallying. Despite these advantages, challenges such as data integrity, unauthorized modification, privacy concerns, and lack of transparency remain major obstacles. Effective e-voting solutions must therefore guarantee that votes are securely stored, verifiable, and resistant to tampering, while maintaining voter anonymity and ease of use.

2.2 Motivation for Blockchain in Voting

Blockchain technology offers inherent properties such as immutability, transparency, and distributed trust, making it highly suitable for secure voting applications. Each block in a blockchain is linked cryptographically to the previous one, preventing unauthorized alterations without detection. This ensures that once a vote is recorded, it cannot be modified or deleted. Blockchain also enables auditability, as every recorded transaction can be independently verified. Traditional e-voting systems rely heavily on centralized databases, which are vulnerable to internal manipulation or external attacks. Incorporating blockchain reduces single points of failure and increases trust in the voting process, especially in controlled environments where full decentralization is not necessary but transparency is critical.

2.3 Proposed Solution: VoteChain

VoteChain is designed as a lightweight blockchain-backed voting platform that operates within a local network. It provides a practical balance between security and simplicity by using a custom blockchain engine to store votes as immutable transactions. The system incorporates role-based access controls, separating administrator and voter operations to ensure proper authorization and election integrity. Administrators can manage candidates, control the election lifecycle, and view real-time results, whereas voters authenticate with their voter IDs and cast exactly one vote. VoteChain's modular architecture, combined with its custom blockchain implementation, ensures transparent vote recording, verifiable tallies, and protection against tampering while remaining easy to deploy and understand.

3 Literature Review

3.1 Existing E-Voting Systems

Traditional electronic voting systems typically rely on centralized architectures in which votes are collected, stored, and counted using a central server or database. Prominent systems include direct-recording electronic (DRE) machines, optical scan systems, web-based e-voting tools, and institutional digital voting frameworks used in universities or corporate elections. These systems aim to improve speed, accessibility, and accuracy compared to paper-based methods. However, multiple studies highlight the challenges they face: susceptibility to tampering, lack of transparency in tallying, insider threats, insufficient audit mechanisms, and potential failures arising from single points of control. Although some systems incorporate cryptographic protocols to enhance privacy or integrity, they often remain vulnerable due to centralized storage and limited verifiability for end users. As a result, trust in purely electronic voting systems remains a significant concern, especially where high-stakes decisions are involved.

3.2 Blockchain-Based Voting Approaches

Recent research explores blockchain technology as a means to enhance transparency, security, and trust in digital voting. Several academic prototypes and pilot systems utilize public or private blockchains to store vote data immutably. Models based on Ethereum, Hyperledger Fabric, and custom Proof-of-Work or Proof-of-Authority chains demonstrate the potential to provide tamper resistance, distributed verification, and auditability without relying on a single authority. These approaches leverage the cryptographic

linking of blocks, consensus mechanisms, and distributed storage to ensure that recorded votes cannot be altered. However, practical deployment remains limited due to challenges such as complexity, high computational or financial cost on public chains, scalability constraints, and the need for secure voter authentication layers that integrate seamlessly with blockchain-backed storage.

3.3 Comparison and Limitations

While blockchain-based voting systems offer stronger protection against tampering than traditional e-voting platforms, many proposed solutions are too complex or resource-intensive for small-scale institutional elections. Public blockchains introduce transaction fees and latency, whereas enterprise-grade blockchains require substantial setup and maintenance efforts. Conversely, conventional e-voting systems are easier to deploy but lack transparency and strong audit mechanisms. These limitations highlight the need for a middle-ground solution: a simple, self-contained blockchain implementation that ensures immutability, verifiability, and clear audit trails without requiring a distributed network or expensive infrastructure. VoteChain addresses this gap by incorporating blockchain principles in a lightweight manner suitable for local-network usage.

4 System Architecture

4.1 Overall Architecture

VoteChain follows a modular, service-based architecture designed to ensure clear separation of concerns and ease of maintenance. The system consists of a FastAPI backend that handles all server-side logic, a custom blockchain engine responsible for secure vote recording, a lightweight SQLite database for storing administrative and voter-related data, and a web-based frontend for administrators and voters. Communication between the frontend and backend takes place via RESTful APIs. The architecture is deliberately kept simple, operating entirely within a local network environment to provide controlled, dependable, and transparent elections.

4.2 Backend Components

The backend is built using FastAPI and encompasses several functional modules: authentication, voter services, admin services, election lifecycle management, and the blockchain engine. The ‘routes‘ layer organizes API endpoints by feature, ensuring clean role separation. The ‘security‘ module manages JWT-based authentication and permission enforcement, preventing unauthorized access. A ‘database‘ module encapsulates models, CRUD functions, and session management. The backend also includes scripts for generating test data and resetting the election, along with a validation and serialization layer for structured request handling. Together, these components form the core operational logic of VoteChain.

4.3 Frontend Interfaces

The frontend is implemented using simple HTML, CSS, and JavaScript, allowing direct interaction with the backend API through asynchronous fetch requests. Two distinct interfaces exist: an administrator dashboard and a voter interface. The administrator interface provides tools for adding and managing candidates, starting or ending elections, viewing voters, monitoring turnout, and inspecting blockchain data. The voter interface allows users to authenticate with their voter ID, view candidates, cast a single vote

per election, and view results after the election concludes. The design prioritizes ease of use and clarity to support users with minimal technical experience.

4.4 Database Schema

The database schema is intentionally straightforward to support efficient management of essential election data. It includes three primary tables: **Voter**, which stores voter identifiers and voting status; **Candidate**, which stores candidate details; and **ElectionState**, which tracks the current election phase (not started, ongoing, or ended). SQLAlchemy is used as the ORM layer, enabling clean abstraction of database operations. While the blockchain remains the immutable record of votes, the database assists in enforcing one-vote-per-user policies and maintaining overall system control.

4.5 Blockchain Flow

The blockchain engine processes vote transactions and stores them in blocks linked by cryptographic hashes. When a voter casts a vote, the system creates a `VoteTransaction` containing a hashed version of the voter ID and the chosen candidate ID. These transactions are temporarily stored in a mempool until they are mined into a block. During mining, a new block is generated containing pending transactions, a timestamp, a reference to the previous block's hash, and its own computed hash. This ensures immutability and verifiability of vote records. Chain validation checks detect any tampering by recomputing and verifying hashes across the entire chain. VoteChain uses a simplified longest-valid-chain rule for conflict resolution, ensuring trust in environments with multiple nodes or instances.

5 Module Description

5.1 Authentication Module

The authentication module is responsible for securely verifying both administrators and voters using JWT (JSON Web Tokens). Administrators authenticate using a predefined password, while voters authenticate with their unique voter ID. After successful login, the system generates a signed JWT token containing the user's role and identity claims. This token must be provided with each subsequent request to authorize protected operations. The module also includes permission enforcement functions that ensure correct role-based access control, blocking unauthorized access to admin or voter routes. The use of JWT allows stateless, efficient authentication without requiring server-side session storage.

5.2 Admin Module

The admin module contains all functionalities required to manage an election from start to finish. Administrators can register, update, or delete candidates, view the list of registered voters, monitor the number of participants, and observe turnout statistics. They are also responsible for initiating and ending the election, which shifts the system's state between inactive, active, and closed phases. Administrators additionally have access to blockchain inspection tools allowing them to view blocks, transactions, and verify transparency of the voting process. This module provides complete supervisory control while maintaining strict role isolation.

5.3 Voter Module

The voter module enables authenticated voters to interact with the system through a sim-

plified interface. Before the election begins, voters can view the list of registered candidates. When the election is active, each voter can cast exactly one vote; the system prevents duplicate submissions by tracking voting status in the database and blockchain layer. After the election ends, voters can view results including vote counts, percentages, and turnout. All sensitive operations are validated server-side to ensure fairness, while voter identities are anonymized through hashing before being recorded on the blockchain.

5.4 Election Lifecycle Module

This module orchestrates the different phases of the election by managing the global election state stored in the database. It supports three states: **NOT_STARTED**, **ONGOING**, and **ENDED**. Transitions are controlled exclusively by the administrator through API endpoints for starting and ending the election. The module enforces consistent behavior across the system: preventing votes before the election starts, allowing exactly one vote per voter during active phases, and exposing results only after closure. It ensures smooth progression of the election and prevents logical inconsistencies such as voting after finalization.

5.5 Blockchain Module

The blockchain module implements a custom lightweight blockchain to securely record votes as immutable transactions. Each vote generates a `VoteTransaction` containing a hashed voter ID and the selected candidate ID. Transactions are stored temporarily until they are mined into a block. The block structure includes timestamp, index, transaction list, previous hash, and a cryptographic hash of its contents. Blocks are linked sequentially to ensure tamper resistance: any modification invalidates the chain. The module also includes chain validation logic and a simple consensus mechanism based on choosing the longest valid chain. This ensures transparency and integrity without requiring complex distributed networks.

6 Implementation Details

6.1 Technologies Used

VoteChain is implemented using a combination of modern web technologies chosen for efficiency, simplicity, and reliability. The backend is built on **FastAPI**, a high-performance Python web framework suitable for RESTful APIs. Database operations are handled using **SQLAlchemy**, with **SQLite** serving as the lightweight relational database for storing voter and candidate information. Authentication relies on **JWT** (JSON Web Tokens) generated using Python's cryptographic libraries. The frontend uses standard **HTML**, **CSS**, and **JavaScript**, ensuring compatibility with any modern browser. A custom **blockchain engine** is implemented in Python to record votes immutably. Testing is performed using **pytest**, enabling automated validation of system modules and workflows.

6.2 Backend Implementation

The backend is organized into modular components to improve maintainability and clarity. API endpoints are grouped into separate route files for authentication, admin operations, voter actions, and election lifecycle control. SQLAlchemy ORM models represent the database schema, and a dedicated CRUD layer manages structured interactions with stored data. The backend enforces role-based access control through custom dependency functions that validate JWT tokens and verify user roles. The overall design follows a clean separation of concerns, allowing

the voting logic, blockchain processing, and data management to operate independently while communicating through well-defined interfaces.

6.3 Frontend Implementation

The frontend consists of lightweight HTML pages enhanced with JavaScript-based API calls for dynamic interaction. Separate interfaces are provided for administrators and voters, each tailored to its functional requirements. Administrators can manage candidates, start or end elections, and review results, while voters can authenticate, view candidates, cast votes, and later inspect final outcomes. The design emphasizes clarity and accessibility, ensuring that users without technical backgrounds can operate the system effectively. AJAX-style requests enable real-time updates without page reloads, resulting in a responsive and user-friendly interface.

6.4 Blockchain Implementation

The blockchain engine is implemented from scratch to demonstrate core principles such as immutability, hashing, and linked block structures. Each vote creates a `VoteTransaction` object, which is temporarily held in a mempool until mining occurs. When a block is mined, it includes the transaction set, timestamp, index, previous hash, and its own computed hash using SHA-256. The engine validates the chain by recomputing and comparing hashes across all blocks. A simple consensus mechanism based on selecting the longest valid chain is included to resolve inconsistencies when multiple chain instances exist, ensuring integrity in controlled network environments.

6.5 Scripts and Utilities

Several helper scripts support system administration and testing. The `generate_test_data.py` script populates the database with sample voters and candidates for demonstration purposes. The `reset_chain.py` script resets the blockchain, election status, and voter activity flags, returning the system to a clean initial state. Shell scripts such as `startup.sh` and `run.sh` streamline launching the backend during development. Additional utilities for validation, serialization, and hashing contribute to cleaner code organization and simplify both backend and blockchain operations.

7 Results and Testing

7.1 Test Scenarios

A comprehensive testing strategy was adopted to validate the functionality, reliability, and security of VoteChain. Test scenarios were designed to cover all critical aspects of the system, including user authentication, admin operations, voting logic, blockchain integrity, and result generation. Both normal and exceptional conditions were evaluated. Scenarios included successful login, failed login attempts, casting valid votes, preventing duplicate votes, validating state transitions during the election lifecycle, and simulating data tampering to confirm blockchain resistance. These scenarios ensured that the system performed correctly under expected use cases and remained resilient against unauthorized actions.

7.2 Unit Tests

Unit tests were used to validate individual components of the backend, ensuring that each module performed its intended function in isolation. Key units tested included authentication token generation, permission enforcement, CRUD operations on voters and

candidates, blockchain hashing, block validation, and chain consistency. The blockchain module, in particular, underwent rigorous unit testing to verify correct block creation, transaction handling, hash computation, and detection of any data manipulation. Unit tests provided confidence that low-level functions behaved reliably before being integrated into the complete workflow.

7.3 Integration Tests

Integration testing focused on validating interactions between multiple modules such as the backend API, database, authentication layer, and blockchain engine. Test cases simulated complete API requests using FastAPI's TestClient to ensure that real-world user actions produced accurate system responses. Integration tests verified that voting transactions were correctly saved, mined, and reflected in the blockchain, and that database updates—for example, marking a voter as having voted—occurred consistently. These tests also ensured seamless communication between frontend requests, backend logic, and persistent storage.

7.4 Voting Flow Validation

End-to-end tests validated the entire voting cycle from start to finish. The workflow included admin login, starting the election, voter login, candidate retrieval, casting a vote, preventing duplicate votes, mining the vote into the blockchain, ending the election, and retrieving final results. Each step was verified for correctness and security. The results confirmed that only authenticated voters could participate, each voter could cast only one vote, and the blockchain accurately recorded all transactions. This validation demonstrated that the platform behaves reliably in practical usage scenarios.

7.5 Performance Analysis

Performance analysis focused on the responsiveness of backend endpoints, transaction processing times, and block creation efficiency. Since VoteChain is intended for controlled, small-scale use, the blockchain design prioritizes simplicity and reliability over heavy computational workloads. Tests showed that API requests consistently executed with low latency, and mining operations completed nearly instantaneously due to the lightweight consensus model. Database interactions were efficient, and the system handled simultaneous requests without noticeable degradation. Overall, performance results demonstrate that VoteChain is well-suited for small elections with fast and predictable response times.

8 Security Considerations

8.1 Threat Model

VoteChain is designed for use in controlled network environments where the primary threats include unauthorized access, vote tampering, data manipulation, impersonation attempts, and local network attacks. The system assumes that users may attempt to cast multiple votes, bypass role restrictions, or alter stored election data. Threats also include potential API misuse, compromised admin credentials, and tampering with stored records. The design accounts for these risks by incorporating strict authentication, blockchain-backed immutability, server-side validation, and clear separation of administrative and voter operations. The system does not assume protection against large-scale distributed attacks but ensures robust safeguards against typical internal and network-level threats.

8.2 Role-Based Access Control

A crucial component of VoteChain's security architecture is its strict role-based access control (RBAC) model. Users are classified as either administrators or voters, and each API endpoint enforces role-specific permissions. Administrators have privileged capabilities such as managing candidates and controlling the election lifecycle, while voters are limited to authentication, viewing candidates, and casting a vote. JWT tokens carry role assignments, and permission checks are implemented at the backend through security dependencies. This approach prevents unauthorized privilege escalation and ensures that sensitive administrative operations remain inaccessible to regular users.

8.3 Blockchain Integrity

The blockchain serves as the core mechanism ensuring vote integrity by storing each vote as an immutable transaction. Once mined into a block, a vote cannot be altered without invalidating the hash chain. Each block contains a SHA-256 hash computed from its contents and the previous block's hash, creating a tamper-evident sequence. The chain validation function continuously verifies block integrity by recomputing and comparing hashes. Any unauthorized changes to stored votes or block data are immediately detectable. Additionally, the simplified longest-valid-chain consensus mechanism ensures resilience in environments where multiple nodes might run independently.

8.4 Privacy and Voter Anonymity

To protect voter privacy, VoteChain never stores voter identities directly on the blockchain. Instead, each vote transaction contains a hashed representation of the voter ID, ensuring anonymity while still allowing fraud detection through one-vote-per-voter enforcement. The database maintains the mapping between voter IDs and their voting status, but this information never appears in blockchain records. No personally identifiable information is transmitted or published, and all communication between the frontend and backend is limited to essential data. This approach balances transparency with confidentiality, ensuring that vote tallies remain verifiable while individual voter choices stay private.

9 Advantages and Limitations

9.1 Advantages

VoteChain offers several advantages that make it suitable for secure and transparent elections in small-scale or controlled environments. Its blockchain-backed storage ensures immutability of votes, preventing unauthorized modifications and providing a verifiable audit trail. The system's modular architecture separates responsibilities clearly, simplifying maintenance and enhancing security. Role-based access control ensures that administrative operations remain strictly restricted, while voters can only perform legitimate actions such as casting a single vote. Additionally, the lightweight nature of the blockchain ensures fast performance without the computational overhead associated with public distributed networks. The web-based interface further enhances accessibility, making the system usable on any device with a modern browser.

9.2 Limitations

Although VoteChain provides strong integrity and transparency, it is not designed for large-scale national elections or high-load dis-

tributed environments. The blockchain implementation is intentionally simplified and does not incorporate advanced consensus mechanisms, distributed replication, or peer-to-peer communication. Since the system operates within a local network, it cannot defend against large-scale external attacks or highly sophisticated adversaries. The reliance on hashed voter IDs offers anonymity but may require additional encryption mechanisms for stricter privacy requirements. Furthermore, the system assumes an honest administrator and does not address insider threats beyond blockchain immutability.

9.3 Possible Improvements

Future enhancements can extend VoteChain's capabilities and robustness. Implementing a fully distributed blockchain with peer-to-peer communication would eliminate single points of trust and increase resilience. Additional security measures such as HTTPS communication, encrypted voter identifiers, and multi-factor authentication could further strengthen the platform. Performance optimizations, such as batched transactions or improved mining strategies, may enhance scalability. Introducing audit dashboards, analytics, or integration with mobile applications could improve usability. Finally, incorporating formal verification or zero-knowledge proof mechanisms could enable stronger guarantees for anonymity and vote correctness.

10 Conclusion

10.1 Summary of Contributions

VoteChain demonstrates how blockchain principles can be applied effectively to create a secure, transparent, and tamper-resistant electronic voting system suitable for small-scale or controlled environments. The project integrates authentication, role-based access control, election lifecycle management, and a custom blockchain engine into a unified platform. Each vote is stored immutably, ensuring strong integrity and auditability. The separation of voter and administrator interfaces enhances usability and security, while the lightweight architecture ensures fast performance without the need for complex distributed networks. Overall, VoteChain highlights the practical use of blockchain technology to strengthen trust in digital voting processes.

10.2 Future Work

Future enhancements can expand the functionality, security, and scalability of VoteChain. Implementing a distributed peer-to-peer blockchain network would remove reliance on a central instance and improve resilience. Additional features such as encrypted voter identifiers, end-to-end verifiable ballots, HTTPS support, and multi-factor authentication could further strengthen privacy and authentication. Expanding the user interface with real-time dashboards or analytics may improve usability for administrators. Integrating more advanced consensus algorithms or zero-knowledge proofs could elevate the system toward more robust and fully verifiable election frameworks suitable for larger-scale deployments.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf, 2008.

[2] M. Swan, *Blockchain: Blueprint for a New Economy*. O'Reilly Media, 2015.

[3] K. Z. Haq, A. Gawanmeh, and S. Mohamad, "A blockchain-based architecture for secure and transparent electronic voting," in *IEEE International Conference on Electronics, Information, and Communication (ICEIC)*, 2020, pp. 1–7.

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

[5] M. Noizat, "Blockchain electronic vote," in *Blockchain 2015 – New Economic Model*, 2015, pp. 1–20.

[6] S. A. A. Rizvi, M. S. Imran, and M. A. Aziz, "A secure blockchain-based e-voting system for environments with limited trust," *IEEE Access*, vol. 9, pp. 40 081–40 091, 2021.

[7] J. Benaloh, "Simple verifiable elections," in *Voting Systems Symposium*, 2006, pp. 1–10.