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 5, Issue 7, May 2025
Impact Factor: 7.67

Blockchain Based Voting System

Prateek Saraswat, Mohit Singh, Kanishka Goswami, Devansh Singhal
CSE (Data Science)
Raj Kumar Goel Institute of Technology, Ghaziabad
09qeprateek@gmail.com, mohit7042singh@gmail.com
Kanishkgoswami12@gmail.com, Devansh06032003@gmail.com

Abstract: *In recent years, the integrity and transparency of electoral processes have come under increased scrutiny. Traditional voting systems, whether paper-based or electronic, often face challenges related to security, voter fraud , data manipulation, and lack of trust. Blockchain technology, known for its decentralized, immutable, and transparent nature, offers a promising solution to these issues. This paper proposes a blockchain-based voting system that ensures secure, transparent, and tamper-proof elections. Utilizing smart contracts and cryptographic techniques, the system enables secure voter authentication, anonymity, and real-time vote tallying, while ensuring that all votes are recorded immutably on a distributed ledger. The proposed system enhances voter confidence, reduces the risk of manipulation, and supports remote voting without compromising security. This approach has the potential to revolutionize democratic participation by providing a trustworthy and efficient electoral process.*

Keywords: Blockchain, Voting System, Java, JavaFX, Decentralized Ledger, Secure Elections

I. INTRODUCTION

The democratic process is based essentially on free, fair, and transparent elections. Conventional voting systems, whether paper-based or electronic, have long been plagued by issues such as tampering, delay in counting, non-transparency, and lack of accessibility. Electronic Voting Machines (EVMs) that are now used in the majority of countries are criticized for being vulnerable to physical tampering and clerical errors, while paper ballot processes entail high expenditures, complexity, and inefficiency in manual counting.

As digital technologies advance, there is an urgent need for more secure and efficient voting systems. Blockchain technology has been touted as a viable solution because of its inherent characteristics: decentralization, immutability, and transparency. A blockchain is a linear chain of blocks, with each block having data and a cryptographic hash of the previous block, so that no historical data can be modified without making the entire chain invalid. This structure naturally resists tampering and fosters trust among participants.

In this project, we are suggesting a blockchain voting system implemented with Java and JavaFX, but without relying on smart contracts or external blockchain platforms. The system emulates blockchain behavior by linking blocks of voting data with cryptographic hashing and secure time-stamping. It ensures that once voted, it will be permanently stored in a manner that cannot be changed without notice.

Our system meets a number of the most critical challenges of traditional voting systems:

Avoiding double voting.

Protecting vote integrity and transparency.

Facilitating secure and anonymous voting procedures.

Minimizing the cost and logistics of elections in general.

Using basic blockchain principles without the overcomplication of smart contracts, this model presents a lightweight, easy-to-understand framework well-suited to educational, institutional, or small-scale democratic applications.

Motivation

The inspiration for this project arises from the growing demand for secure, open, and reliable voting systems in the digital age. Elections are the cornerstone of any democracy, yet the past few years have identified wide gaps in current voting systems—from claims of electoral manipulation and fraud to reduced access for some voters.

Mechanical paper-based electoral systems are frequently slow, costly, and error-prone due to human interference, whereas Electronic Voting Machines (EVMs), as much more contemporary a system, are still vulnerable to physical manipulation and are non-audit-able. Such limitations create uncertainty among the general populace, diminish the confidence of the voters, and undermine the legitimacy of democratic processes.

Blockchain technology, with its own characteristics of immutability, decentralization, and transparency, offers a strong alternative. While widely linked to cryptocurrencies, its applicability in other areas—such as voting—is increasingly being realized. A voting system based on blockchain can guarantee that once a vote is made, it is safely stored and cannot be altered or erased, thus maintaining electoral integrity.

This project intends to emulate the fundamental concepts of blockchain to show how records of votes can be safely stored without the need for third-party infrastructure or intricate smart contracts. Using the system on Java and JavaFX, we make it available and easy to comprehend, thereby making it a feasible teaching aid and proof of concept for institutional voting applications like college elections, organizational surveys, or local community decision-making.

Our mission is to demonstrate a proof-of-concept that illustrates how technology can bring about trust in voting systems by increasing their security, efficiency, and ease of use—particularly for remote and digitally networked citizens.

Objectives

The main purpose of this project is to plan and develop an secure, open, and tamper-evident voting system based on the fundamental principles of blockchain technology—without relying on smart contracts. Through Java-based simulation of blockchain behavior, the system hopes to increase the trust, usability, and integrity of the election process, especially for small-scale or institutional elections.

The major aims are:

1. To Ensure Vote Integrity

Ensure that each vote is stored permanently and cannot be altered, deleted, or manipulated, thus maintaining the integrity of the election process.

2. To Simulate Blockchain Functionality

Use blockchain-like functionality like chaining of data blocks and cryptographic hashing in Java, without depending on actual blockchain platforms or smart contracts.

3. To Prevent Double Voting

Use mechanisms that enable each user to vote once, maintaining fairness and accuracy.

4. To Maintain Voter Anonymity

Make sure that although every vote is registered and auditable, the voter's identity is kept confidential to ensure privacy.

5. To Improve Transparency and Auditability

Make the whole voting and result calculation process auditable so that administrators and voters have confidence in the results.

6. To Simplify Election Management

Offer an admin interface for voter management, election initiation, and result tracking, thus lessening reliance on manual operations.

7. To Enhance Accessibility and Usability

Develop a simple user interface through JavaFX so that the system can be easy to use even for less-technical users.

8. To Cut Election Expenses and Resources

Provide a virtual option that reduces the demand for tangible resources such as polling stations, paper ballots, and staff.

II. SYSTEM ARCHITECTURE

1. User Interface (UI) - JavaFX

Role: The user interface (UI) is developed using JavaFX, which offers a rich set of tools for building graphical interfaces in Java. It enables you to build windows, buttons, tables, forms, and other interactive elements.

Components:

- Controllers: JavaFX uses the MVC (Model-View-Controller) pattern, where the view (UI) is decoupled from the logic (Model). The controllers handle user input, update the view, and communicate with the back-end services.
- FXML: You can declare the UI layout using FXML, an XML-based format. This enables you to decouple the UI from the code logic.
- Scene Graph: JavaFX scene graph is utilized to position UI elements, i.e., buttons, labels, text fields, etc., in a tree form for simple layout management.

2. Backend Logic - Java

Role: Java is responsible for the application's core logic, i.e., blockchain operations, transactions, and database.

Blockchain Implementation:

The blockchain can be implemented in Java with the help of custom classes that mimic block generation, transactions, and chaining.
Smart contracts (in Solidity) can talk to the blockchain but must be incorporated through Web3j or an equivalent Java library.

Blockchain Functions:

- Block: Every block has a collection of transactions, the hash of the previous block, and its hash.
- Chain: The blocks are connected, and they make up a chain. Every new block has the hash of the old block to keep the whole chain intact.
- Consensus Mechanism: You could use a simple consensus mechanism to confirm transactions (proof-of-work or a simpler one depending on your needs).
- Smart Contract Interaction: Smart contracts, programmed in Solidity, communicate with the blockchain network. Through libraries such as Web3j, Java can communicate with the Ethereum blockchain to install and run smart contracts.

3. Database - SQLite

Role: SQLite persists application data. It's a serverless, lightweight, and self-contained relational database engine.

Integration:

- SQLite Driver: You can use an SQLite JDBC driver to access the database from Java. The driver enables you to create tables, insert, update, delete records, and execute SQL queries.
- Database Structure: SQLite might be employed in a blockchain application for storing metadata related to blocks, transactions, and user data. Though the chain and the transactions are stored within the blockchain itself, SQLite would be more appropriate for storing user-specific information (such as wallet addresses, authentication).

Database Operations

When an action is performed by a user (e.g., voting), the transaction is recorded in the blockchain and may also be stored in the SQLite database for historical purposes, user management, etc.

4. Workflow Overview

1. UI Layer (JavaFX):

- The user interacts with the UI, e.g., login, voting, transaction history, etc.
- JavaFX talks to the backend to get the corresponding actions.

2. Backend Layer (Java):

- When a user makes an action, the backend processes it. If it's an action related to blockchain (e.g., voting), Java will form a transaction, verify it, and include it in the blockchain.
- Blockchain logic verifies the transaction through the consensus mechanism and adds it to the chain.
- The smart contract (if used) runs and sends back the output.

3. Database Layer (SQLite):

- SQLite holds all user-specific data or metadata, like login credentials of a user, transaction records, or other context-specific application data.
- The database is persisted as the application takes actions to ensure persistence of data across sessions.

4. Transaction Flow (Example: Voting System)

- User Interaction (JavaFX): A user accesses the voting page in the UI.
- Backend Logic (Java): The system verifies whether the user is qualified to vote, fetches the candidates available, and enables the user to vote.
- Blockchain (Java): The vote is considered a transaction in the blockchain. A new block is formed with the voting details and appended to the chain.
- Database (SQLite): The system records the vote of the user and updates the database for tracking.
- Smart Contract (Solidity): In case the blockchain contains a smart contract, it will be invoked to validate or process the vote.
- Confirmation (JavaFX): The user gets confirmation within the UI that the vote succeeded.

5. Security Considerations

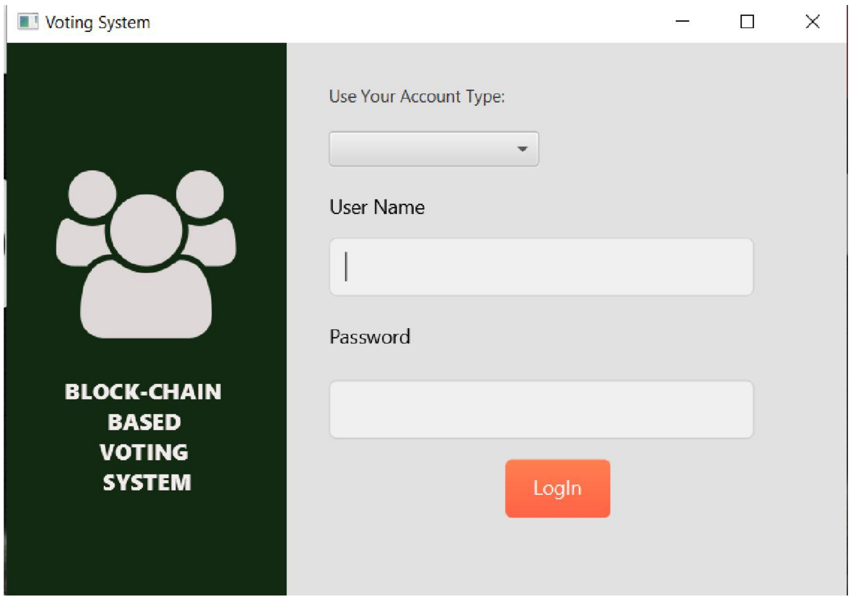
- Blockchain Security: The blockchain guarantees that once a transaction (such as a vote) has been added, it's tamper-proof and traceable because of the cryptographic hash functions employed.
- Database Security: As SQLite holds sensitive information, ensure there is proper encryption and secure management of user credentials and other personal information.
- Smart Contract Security: In case of using smart contracts, make sure they are well audited to avoid issues like reentrancy attacks or overflow problems.

III. IMPLEMENTATION DETAILS

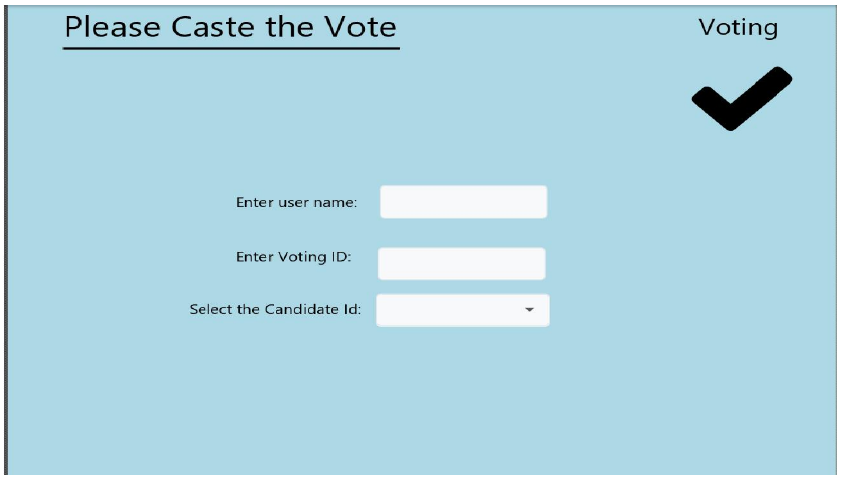
3.1 JavaFX UI Component

- Login Screen (Voter Authentication)

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 5, Issue 7, May 2025
Impact Factor: 7.67



- Voting Dashboard



3.2 Java Backend Components

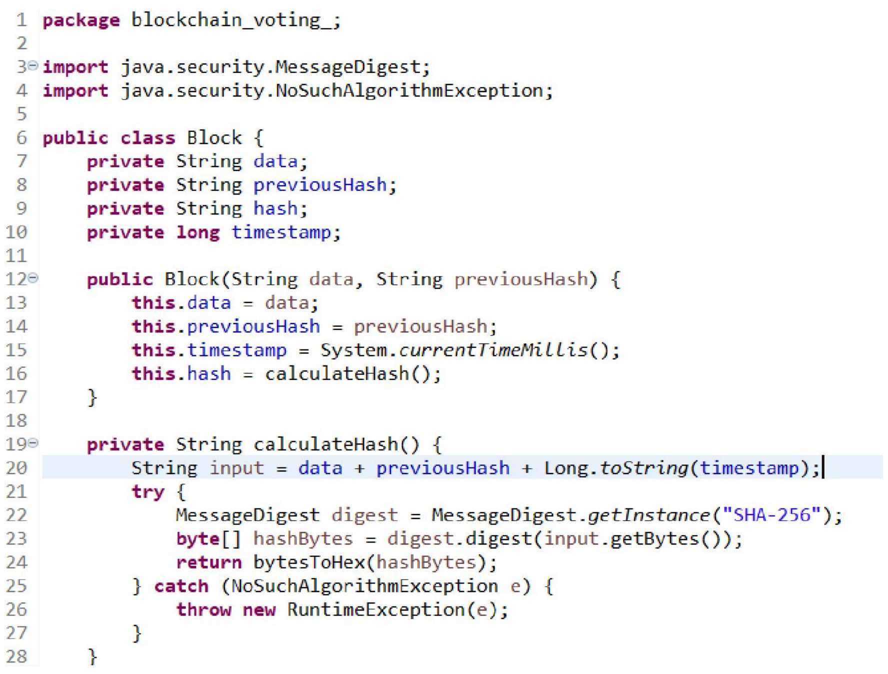
- Blockchain Structure

The Block class is an instance of a piece of data on the blockchain. A block carries the following attributes:

Block Class

- Data: This is what the block contains, which could be encrypted ballots or voting information in the context of a voting system.
- Previous Hash: The hash of the previous block using cryptography, thereby creating a reference between blocks.
- Hash: A unique identifier of the current block, calculated with the SHA-256 hashing function.
- Timestamp: The creation time of the block, thereby allowing chronological ordering.

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 5, Issue 7, May 2025
Impact Factor: 7.67

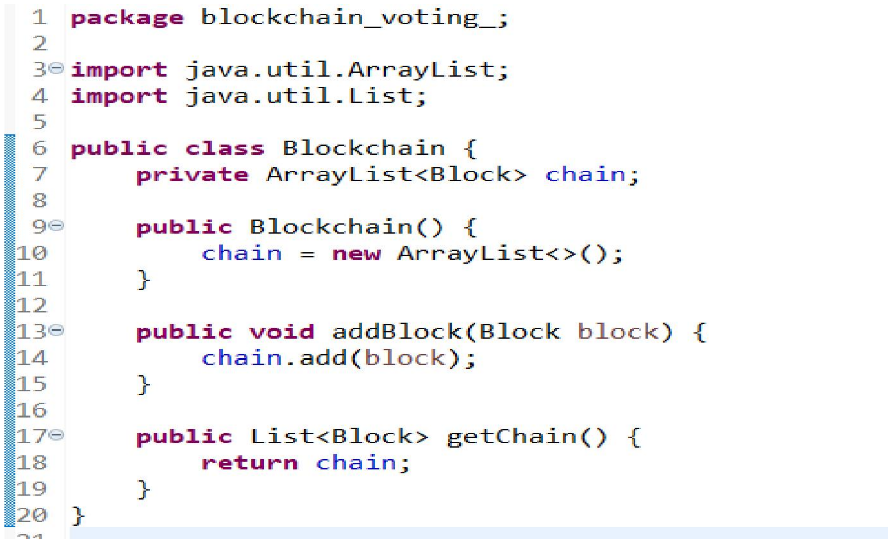

```
package blockchain_voting_;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class Block {
    private String data;
    private String previousHash;
    private String hash;
    private long timestamp;

    public Block(String data, String previousHash) {
        this.data = data;
        this.previousHash = previousHash;
        this.timestamp = System.currentTimeMillis();
        this.hash = calculateHash();
    }

    private String calculateHash() {
        String input = data + previousHash + Long.toString(timestamp);
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] hashBytes = digest.digest(input.getBytes());
            return bytesToHex(hashBytes);
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }
```

Blockchain Class

The Blockchain class is the ledger that holds an ordered sequence of blocks. Internally, it has an ArrayList holding the sequence of blocks:

addBlock(Block block): This function adds a new block to the chain.

getChain(): This function returns the current blockchain state, with exposure to the whole sequence of blocks recorded.


```
package blockchain_voting_;

import java.util.ArrayList;
import java.util.List;

public class Blockchain {
    private ArrayList<Block> chain;

    public Blockchain() {
        chain = new ArrayList<>();
    }

    public void addBlock(Block block) {
        chain.add(block);
    }

    public List<Block> getChain() {
        return chain;
    }
}
```

-Voting System

The Voting System class consolidates two main pieces:

Blockchain Ledger: A distributed ledger through the Blockchain class, in which all vote history is stored safely in cryptographically chained blocks.

Registered Voter List: An aggregate of valid voters kept in a list of Voter objects, which enforces that only eligible persons are able to vote in the election.

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 5, Issue 7, May 2025
Impact Factor: 7.67

IV. SECURITY AND PRIVACY ISSUES

Security and privacy are the pillars of any electoral system. Although blockchain provides a basis of immutability and openness, the system needs to be implemented with caution so that it stays secure from threats while being anonymized. This Java-built project, which also incorporates JavaFX and SQLite, deals with a number of security and privacy issues, though some limitations are unavoidable due to its simulated nature.

4.1 Data Integrity and Tamper Detection

Every vote in the system is considered a block, containing a timestamp, candidate ID, anonymized voter ID, and a SHA-256 hash referencing the previous block. This hash chaining makes it such that:

- Any change to a prior vote instantly breaks the hash chain.
- Tampering becomes traceable via hash verification.
- This structure ensures data immutability, an essential requirement for secure voting.

4.2 Authentication and Authorization

Role-Based Access: Functionality is limited by roles (voter or admin), reducing unauthorized access.

Login System: Admins and voters need to log in with credentials safely stored in SQLite. This does not allow external users to vote or access administrative features.

Single Vote Enforcement: The system guarantees that a voter can only cast one vote, monitored in the database and checked before granting access to the voting panel.

4.3 Voter Anonymity

Though vote data is kept and presented to be transparent, the voter's identity is anonymized by:

- Using internal IDs rather than personal identifiers.
- Blockchain blocks that do not hold identifiable information directly.
- This way, votes cannot be traced back to individuals without violating their privacy while ensuring system auditability.

4.4 Database Security

Although SQLite is light and suitable for prototypes, it does have some limitations regarding encryption and multi-user security:

- Local File Storage: SQLite stores data within a local file that may be at risk if not secured through OS-level permissions or encryption.
- No Network-Level Security: As the database is not hosted by a server, there is no inherent protection against network attacks such as SQL injection or sniffing. This is addressed by input sanitization and limiting access to the application level.

4.5 Risks and Limitations

Even with the security controls in place, the system has some limitations:

No End-to-End Encryption: Contrary to actual blockchain implementations, this project lacks encryption from user input to data storage, which may pose a threat when scaled up.

Centralized Storage: Although integrity is added in simulation through blockchain, real-world storage is centralized using SQLite, presenting a point of failure.

No Biometric or 2FA Authentication: The existing login system relies only on passwords, which can be vulnerable to brute-force or credential steal attacks.

4.6 Recommendations for Increased Security

To further enhance the system, the following enhancements are suggested for future development:

- Utilization of encrypted databases or external secure servers.

- Two-factor authentication (2FA) for login.
- Encrypt votes prior to storage, even in hashed form.
- Migration to decentralized blockchain platforms for real-time integrity and network-level trust.
- Regular audits and hash checks to verify vote chain integrity.

V. CHALLENGES AND SOLUTIONS

Development and deployment of a blockchain voting system pose some technical, operational, and user-oriented challenges. Although the project uses Java, JavaFX, and SQLite to mimic blockchain operations, there are tremendous challenges in attaining solid performance, security, and usability. The following are the major challenges faced and the solutions or mitigation measures employed:

5.1 Challenge: Lack of True Decentralization

Problem:

Actual blockchain systems function in a decentralized framework where more than one node verifies and maintains the ledger. This simulation employs a centralized scheme, which can't possibly reflect blockchain's distributed-based nature.

Solution:

To mimic decentralization, the system employs immutable hash chaining and locally stores every vote block sequentially. While not actually decentralized, this replicates the tamper-proof aspect of a blockchain and is an educational model. Networked nodes or integration with platforms such as Hyperledger could be included in future development.

5.2 Challenge: Ensuring Vote Anonymity

Problem:

Maintaining voter anonymity while having verifiable election results is essential. Directly linking votes to user information threatens anonymity.

Solution:

The system anonymizes the voter data prior to supplementing vote data on the blockchain. Every block contains only necessary, non-identifiable details (e.g., timestamp of the vote, candidate ID) and employs internal IDs in order to avoid disclosing identity.

5.3 Challenge: Double Voting Prevention

Problem:

A user may try to vote twice, particularly in electronic systems without strong ID verification.

Solution:

The system maintains the voting status of every user in the SQLite database. After a vote is cast, the record of the voter is flagged, and subsequent attempts are prevented at the application level.

5.4 Challenge: Database Vulnerability

Problem:

SQLite, though light, does not have native encryption and is exposed if the local machine is compromised.

Solution:

Application-level security is implemented with access restrictions, hashed passwords, and file-level OS protection. Future improvements may include encrypted databases or migration to secure server-based storage.

5.5 Challenge: Limited User Technical Literacy

Problem:

Some voters may not be familiar with digital systems, leading to confusion or errors during voting.

Solution:
A JavaFX interface that is simple and easy to understand is employed for simplicity. The interface has easy-to-follow instructions, guided processes, and validation of errors to enhance the user experience.

5.6 Challenge: No Live Network Validation

Problem:
An actual blockchain network provides consensus mechanisms to confirm entries, something that does not exist in the simulation here.
Solution:
Although consensus protocols (such as Proof-of-Work or Proof-of-Stake) are not used, the protocol is based on local verification and hash verification to guarantee block integrity. This model is appropriate for proofs of concept on the blockchain at a prototype level.

5.7 Challenge: Timing Delayed Result Accuracy and Real-Time Monitoring

Problem:
Manual vote tallying or insecure tallying mechanisms can lead to inaccuracy or timing issues.
Solution:
The system carries out real-time tallying of votes with data consistency verification between the database records and blockchain ledger. Results are obtainable immediately once voting closes, minimizing delays.

VI. FUTURE WORK

Although the existing implementation of the blockchain-based election system gives a working example of secure and tamper-free elections with Java, JavaFX, and SQLite, there is vast room for improvement. Future work and research activities can focus on removing current constraints and enhancing system capabilities for more extensive usage and scalability. Some important areas of future work are:

6.1 Integration with Real Blockchain Networks

The current model emulates blockchain with hash-linked data structures. Future releases may incorporate actual blockchain platforms like:
- Hyperledger Fabric for permissioned networks for governmental use.
- Ethereum in case of later reintroduction with support for smart contracts.
- This would enable actual decentralization, distributed consensus, and secure networked transactions.

6.2 Advanced Cryptographic Techniques Implementation

Data security and anonymity of users can be strengthened with:
- Zero-Knowledge Proofs (ZKPs) to authenticate votes without disclosing the identity of the user.
- Elliptic Curve Cryptography (ECC) for secure, lightweight encryption.
- Homomorphic encryption for vote tallying without decrypting votes.

6.3 Biometric and Multi-Factor Authentication

In order to enhance user authentication, the future systems can include:
- Biometric authentication (fingerprint or facial recognition).
- Two-factor authentication based on OTPs or email-based codes.
This would improve identity verification without affecting user experience.

6.4 Mobile Application Development

Creating a mobile app for Android and iOS operating systems using technologies such as Kotlin, React Native, or Flutter would:

- Make it more accessible, particularly in remote locations.
- Promote wider usage among youth and technology-savvy voters.

6.5 Cloud and Distributed Hosting

Running the system over a cloud-based platform (e.g., AWS, Google Cloud, Azure) could:
- Enable large-scale elections.
- Increase availability, scalability, and fault tolerance.
- Facilitate remote backups and real-time data synchronization.

6.6 Government and Legal Integration

Subsequent versions must comply with:
- National election commission regulations
- Data privacy laws (e.g., GDPR, IT Act).
- Digital signature legislation to facilitate legal acceptance of digital votes.

6.7 Usability Testing and Voter Education

One of the most important features of real-world deployment is that:
- The system is accessible for all age groups.
- Citizens are properly educated on digital voting.
- Future releases may incorporate interactive tutorials, FAQs, and demo platforms to enhance digital literacy and uptake.

6.8 Real-Time Monitoring and Analytics

Adding dashboards for:
- Live tracking of turnout.
¬- Vote analytics by candidate.
Geographic heat maps of votes based on anonymized data.
This would increase transparency in elections and administrative decision-making.

VII. CONCLUSION

This project investigates the creation of a blockchain-based voting system aimed at securing electoral processes, increasing transparency, and improving efficiency. Using Java for core logic implementation and simulating the blockchain, JavaFX for an interactive graphical user interface, and SQLite for lightweight but reliable data storage, the system offers a robust proof-of-concept for secure digital voting. The basic blockchain principles—such as immutability, hashing, and block chaining—are replicated successfully to preclude vote tampering and enable traceability of votes.

The system overcomes significant weaknesses of conventional and electronic voting systems, including susceptibility to manipulation, absence of transparency, high cost of operation, and inaccessibility. The data of voters is secured using cryptography methods, while the integrity of the election process is ensured through imposing one vote per user and ensuring that each vote is tamper-proof recorded. While the system is yet to use smart contracts or total network decentralization, it does set a strong groundwork for upcoming advancements and application in the real world.

This study also points to the significance of establishing trust in electoral systems using technology. Through the simulation of blockchain behavior even within a centralized system, the project illustrates how verifiable and transparent elections can be performed online. Additionally, the system can accommodate remote voting, thus making it more accessible to individuals with physical disabilities, those located far away from the city or town, or citizens abroad.

In summary, the suggested blockchain-based voting system is an important milestone toward reimagining election processes in the modern digital age. Not only does it enhance the integrity and effectiveness of voting, but also enhances voter engagement and confidence. Integration with existing decentralized blockchain networks in the future, biometrics, and mobile applications may produce a scalable, production-grade solution for conducting mass-scale, nationwide elections.

REFERENCES

[1]. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf

[2]. Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2018). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. IEEE Congress on Big Data, 557–564.

[3]. Crosby, M., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain Technology: Beyond Bitcoin. Applied Innovation Review, 2, 6–10.

[4]. Christin, N. (2020). The Future of Secure Voting: Blockchain and Beyond. Communications of the ACM, 63(4), 28–30.

[5]. Sharma, T., & Agarwal, R. (2019). Blockchain Technology in E-Governance and Voting. International Journal of Computer Applications, 178(7), 1–4.

[6]. Yavuz, E., Koç, A. K., Çabuk, U. C., &Dalkılıç, G. (2018). Towards Secure E-Voting Using Ethereum Blockchain. ISDFS Proceedings.

[7]. JavaFX Documentation. Oracle. https://openjfx.io

[8]. Kshetri, N. (2017). Can Blockchain Strengthen the Internet of Things? IT Professional, 19(4), 68–72.

[9]. Szabo, N. (1997). The Idea of Smart Contracts. https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html

[10]. Rikken, O., &Zwitter, A. (2020). Trust in Blockchain-Based Voting Systems: The Role of Public Perception. Technology in Society, 63, 101395.

[11]. Ali, R., Clarke, N., & Furnell, S. (2020). An Evaluation of the Usability and Security of E-Voting Systems. Computers & Security, 88, 101640.

[12]. Ayed, A. B. (2017). A Conceptual Secure Blockchain-Based Electronic Voting System. IJNSA, 9(3), 1–9.

[13]. Atzori, M. (2017). Blockchain Technology and Decentralized Governance. Journal of Governance and Regulation, 6(1), 45–62.

[14]. Khurana, M., & Baral, R. (2021). A Secure E-Voting System Using Blockchain and Fingerprint Authentication. Procedia Computer Science, 185, 318–326.

[15]. Kim, H. M., & Laskowski, M. (2018). Ontology-Driven Blockchain Design for Provenance. ISAFM, 25(1), 18–27.