

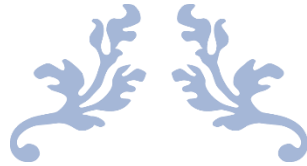
CCT College Dublin

Assessment Cover Page

| | |
|-----------------------------|---|
| Module Title: | Distributed Digital Transactions |
| Assessment Title: | CA1 Project |
| Lecturer Name: | Dr. Muhammad Iqbal |
| Student Full Name: | Guilherme Felix & Leonardo Oliveira |
| Student Number: | 2021309 & 2021361 |
| GitHub Link: | https://github.com/guifeliix/DDT-CA1 |
| Video Link: | https://drive.google.com/file/d/1tTf911qn2KNYnnyUUS0qXRxUjFn3w61S/view?usp=sharing |
| Assessment Due Date: | 03/12/2023 |
| Date of Submission: | 05/12/2023 |

Declaration

By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.



NEW WORLD ORDER, ONE ELECTION AT A TIME

Our Blockchain Voting System



DECEMBER 5, 2023

CCT DUBLIN

Guilherme Felix & Leonardo Oliveira

Contents

| | |
|--|----|
| Abstract: | 3 |
| 1. Introduction | 3 |
| 1.1 Background | 3 |
| 1.2 Literature Overview | 3 |
| 1.2.1 Evolution of Voting Systems | 3 |
| 1.2.2 Prevalent Issues with Current Voting Systems | 4 |
| 1.3 Need for Innovation | 4 |
| 1.3.1 Vulnerability to Tampering: | 4 |
| 1.3.2 Lack of Transparency: | 4 |
| 1.3.3 Resource-Intensive Processes: | 4 |
| 2. Objectives | 5 |
| 2.1 Enhancing Transparency | 5 |
| 2.2 Ensuring Voter Privacy | 5 |
| 2.3 Reducing Vulnerability to Manipulation | 5 |
| 2.4 Providing a Cost-Effective and Secure Platform | 6 |
| 3. Proposed System General Architecture | 6 |
| 3.1 Overview | 6 |
| 3.2 How our prototype work | 7 |
| 4. Diagrams | 13 |
| 5. Economic Advantages | 15 |
| 5.1 Cost Analysis | 15 |
| 6. Conclusion | 15 |
| Bibliography: | 16 |
| GitHub-Link: | 16 |
| Video-Link: | 16 |

Abstract:

The proposed Blockchain based electronic voting system aims to transform traditional voting methods by improving security, transparency, and efficiency. This paper goes over an overview of the system's architecture, goals, and a detailed examination of the current voting landscape.

1. Introduction

Blockchain technology, known for decentralization and security, is transforming voting systems by addressing issues like security, transparency, and accessibility. This paper explores Blockchain based voting systems, leveraging features such as transparency and decentralized control to overcome challenges in traditional voting. Conventional systems face scepticism due to security and trust issues, highlighting the need for innovative alternatives like Blockchain. The paper discusses current voting system shortcomings, emphasizing vulnerability to hacking and lack of transparency. It introduces a proposed system combining Blockchain with a client-server architecture to enhance security and transparency. Blockchain based voting systems have the potential to reshape elections, introduce trust, and set new benchmarks for secure and reliable voting practices through decentralization and transparency.

As the globe grapples with changing expectations of democratic processes, Blockchain based voting systems represent a paradigm leap that has the potential to reshape the way elections are conducted. This approach has the ability to introduce trust in voters, enhance the democratic basis, and create a new benchmark for safe and dependable voting practises by utilising the ideas of decentralisation, encryption, and transparency.

1.1 Background

Voting systems have evolved from paper ballots to electronic voting machines (EVMs). This section briefly introduces this historical context.

Before the advent of EVMs, ballot papers, resembling small balls or pieces of paper, were frequently used for secret voting. Originating from the Italian word "ballotta," meaning a little ball used in voting, this practice traces back to Venice, Italy. Voters indicated their choices on printed ballots, which were then totalled and saved for potential challenges. In India, palm leaves were historically used for town gathering decisions. These leaves, known as Kudavolai, had candidates' names inscribed and were counted after voting. However, paper ballots, despite being a fundamental aspect of basic elections, faced challenges like fraudulent voting and booth capturing.

1.2 Literature Overview

1.2.1 Evolution of Voting Systems

Traces the historical evolution of voting systems, highlighting the transition to EVMs and other technological advancements.

Paper ballots were the norm before the adoption of EVMs. In India, for example, the shift to EVMs in the 1990s addressed issues like fraudulent voting and booth capturing. The state-owned Electronics Corporation of India and Bharat Electronics played a crucial role in developing and testing EVMs, gradually introducing them in Indian elections from 1998 to 2001.

In Brazil, EVMs have been in use since 1996, when it was first implemented in Santa Catarina's State election. In the general election of 1998, more than 50% of the election was done using EVMs. By the 2000's all elections were being run using electronic voting machines. Brazil is today a good example of a large democracy that relies entirely on technology to run its elections.

1.2.2 Prevalent Issues with Current Voting Systems

Challenges with paper ballots included fraudulent voting and booth capturing due to the manual counting process. The introduction of EVMs in India and Brazil have helped reduce electoral fraud, booth capturing, and improved the competitiveness of elections. Although EVM's have largely improved countries electoral systems, it is still faced with questions by part of the public. The risk of hacking prevails and the need to improvement is constant. Electronic system can still be tampered with, and when there's a need to audit casted votes, the whole device needs to be switched off.

1.3 Need for Innovation

The need for innovation in voting systems arises from vulnerabilities in traditional paper-based methods, particularly those relying on physical paper ballots.

1.3.1 Vulnerability to Tampering:

Paper ballots are susceptible to manual alterations and fraudulent activities like ballot stuffing, compromising election fairness. Verification challenges, like verifying the authenticity of each paper ballot making it a cumbersome process, further contribute to the vulnerability.

1.3.2 Lack of Transparency:

Traditional systems lack transparency, as the journey of a paper ballot is often obscure. Added to that we have the opaque nature of paper ballots, which contributes to an environment where verifying the accuracy of the election results becomes challenging. In addition, manual counting processes, while prone to errors, lack transparency and raise legitimacy questions.

1.3.3 Resource-Intensive Processes:

Added to the issues of vulnerability and transparency, traditional paper-based voting involves time-consuming counting, high printing costs, and logistical challenges, leading to inefficiency and increased expenses.

Because of that, there's a need for new solutions to address weaknesses and inefficiencies in traditional voting systems, like traditional paper-based voting systems' weaknesses, lack of transparency, and resource-intensive nature, as well as the difficulties of verifying EVMs on the spot. The proposed Blockchain based voting system offers a potential revolution by providing a secure, transparent, and efficient democratic process.

2. Objectives

Our proposed blockchain voting system aims to address current voting system limitations while leveraging Blockchain advantages.

2.1 Enhancing Transparency

One of the primary goals of the blockchain based system is to improve the transparency of the entire voting process. Leveraging the decentralized and distributed nature of blockchain, the system aims to provide a full and publicly accessible ledger of all transactions.

- Utilizing blockchain's decentralized nature to provide an immutable transaction history, ensuring trust in the electoral process.
- Enabling public verification of election results through open access to the blockchain, acting as a deterrent against fraudulent activities.

2.2 Ensuring Voter Privacy

We know transparency is crucial but alongside that our blockchain based system has a strong emphasis on maintaining the privacy of individual voters. Our system incorporates cryptographic techniques to secure the identity and choices of voters.

- Incorporating cryptographic techniques to encrypt votes, safeguarding individual choices and identities.
- Implementing decentralized identity verification to enhance voter privacy by distributing verification across the blockchain network.

2.3 Reducing Vulnerability to Manipulation

Another advantage of a blockchain is its resistance to tampering or unauthorized alterations. Our system leverages this characteristic to reduce the vulnerability of the electoral process to manipulation.

- Leveraging blockchain's resistance to tampering to reduce the vulnerability of the electoral process.
- Using smart contracts for secure transactions, automatically validating and enforcing election rules to minimize manipulation.

2.4 Providing a Cost-Effective and Secure Platform

Our blockchain voting system wants to introduce efficiency and cost-effectiveness into the electoral process, mitigating the issues around the laboursome intensive aspect of traditional voting systems.

- Streamlining election processes through automation, reducing time and resources required for voter registration and counting.
- Eliminating printing costs associated with physical ballots, resulting in significant cost savings.
- Enhancing security through blockchain's inherent features, such as decentralized control and cryptographic verification, to minimize fraud and manipulation risks.

In summary, our Blockchain voting system wants to completely redesign the electoral landscape by introducing key objectives: enhancing transparency, ensuring voter privacy, reducing vulnerability to manipulation, and providing a cost-effective and secure platform for elections.

3. Proposed System General Architecture

3.1 Overview

The system that we proposed would combines Blockchain technology with a client-server architecture, facilitating secure and transparent elections. On this paper will be present a prototype of a system that uses blockchain technology to address and mitigate all issues that have been raised with traditional voting systems. Thus, the system caters to both individual voters using smartphones and computers and large-scale elections with electoral voting machines connected to a Blockchain network.

Though developers have envisioned a variety of ways to deploy Blockchain technology during voting, there are a few key procedures that many believe would occur in a Blockchain system. The following would probably happen in a decentralised voting process:

1. Voters need to provide identity in order to cast a ballot.
2. All data is stored in transactional form and is encrypted.
3. The transaction is broadcast to every node in the network, and it is quickly verified.
4. The data is stored in a block and added to the chain when the network approves the transaction; once included, it cannot be removed or changed.
5. Users may see the results and, if they'd want, review earlier transactions.

3.2 How our prototype work

1. Contract Initialization:

- The contract is named `Voting` and specifies the compiler version and licensing.
- The `pragma experimental ABIEncoderV2;` is used to enable experimental features related to ABI encoding.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3 pragma experimental ABIEncoderV2;
4
5 contract Voting {
6     // Admin address who manages the election
7     address public admin;
8     // Voter structure
9     TheVoter[] private TheVotersList;
10    mapping(address => TheVoter) private TheVoters;
11}
```

2. Admin and Voter Structures:

- The contract has an `admin` address, representing the administrator who manages the election.
- There is a `TheVoter` structure to store information about voters, including name, registration status, voting status, and address.
- An array `TheVotersList` and a mapping `TheVoters` are used to keep track of registered voters.

```
6 // Admin address who manages the election
7 address public admin;
8 // Voter structure
9 TheVoter[] private TheVotersList;
10 mapping(address => TheVoter) private TheVoters;
11
12 // representatives structure
13 struct TheCand {
14     string name; // Name of the rep
15     uint256 votes; // Number of votes received by the rep
16     bytes32 hash; // Hash of the rep's name
17 }
18 struct TheVoter {
19     string name; // Whos voting
20     bool isRegistered; // Flag indicating if the voter is registered
21     bool hasVoted; // Flag indicating if the voter has voted
22     address TheVoterAddress; // Address of the voter
23 }
24 TheCand[] public representativesList; // List of rep
25
26 // Voting statistics
27 uint public allVotesIn; // Total allVotes
28
29 // Constructor, sets the admin address
30 constructor () { 1425331 gas 1403000 gas
31     admin = msg.sender;
32 }
```


3. Representative Structure:

- There is a `TheCand` structure to store information about representatives (candidates), including name, votes received, and a hash of the representative's name.
- An array `representativesList` is used to store the list of representatives.

```
// representatives structure
struct TheCand {
    string name;// Name of the rep
    uint256 votes;// Number of votes received by the rep
    bytes32 hash;// Hash of the rep's name
}
```

```
TheCand[] public representativesList;// List of rep
```

4. Voting Statistics:

- A variable `allVotesIn` is used to keep track of the total number of votes cast.

```
// Voting statistics
uint public allVotesIn; // Total allVotes
```

5. Voter Registration:

- The contract includes a function `registerTheVoter` that allows only the admin to register voters.

```
// Register TheVoter -- only admn can
function registerTheVoter(address _TheVoter, string memory _name) external {  infinite gas
    require(msg.sender == admin, "Adminitrator function");
    require(_TheVoter != address(0), "Not a TheVoter add");
    TheVoters[_TheVoter] = TheVoter(_name, true, false, _TheVoter);
    TheVotersList.push(TheVoters[_TheVoter] );
}
```


6. Getting Voter Details:

- The function `getTheVoterDetails` is provided to retrieve details of a registered voter.

```
// TheVoter details
function getTheVoterDetails(address _TheVoter) private view returns (string memory) {  in
    require(TheVoters[_TheVoter].isRegistered, "TheVoter is not in memory");
    return TheVoters[_TheVoter].name;
}
```


7. Checking if Voter Voted:

- The function `CheckIfVoterVoted` checks whether a voter has cast their vote.

```
// Check did TheVoter voted
function CheckIfVoterVoted(address _TheVoter) private view returns (bool) {  infinite gas
    return TheVoters[_TheVoter].hasVoted;
}
```


9. Representative (Candidate) Registration:

- The contract includes a function `addTheCand` that allows only the admin to add representatives (candidates).

```
// Put representatives on system -- admin can
function addTheCand(string memory _name) external {  infinite gas
    require(msg.sender == admin, "Adminitrator function");
    require(bytes(_name).length > 0, "TheCand has to be filled");
    bytes32 hash = keccak256(bytes(_name)); // create a hash for representatives
    representativesList.push(TheCand(_name, 0, hash));
}
```


10. Getting List of Representatives:

- The function `getTheCands` retrieves an array of names of all representatives.

```
// Get the names of all reps
function getTheCands() private view returns (string[] memory) {  infinite gas
    string[] memory representativesNames = new string[](representativesList.length);
    for (uint256 i = 0; i < representativesList.length; i++) {
        representativesNames[i] = representativesList[i].name;
    }
    return representativesNames;
}
```

11. Getting List of Representatives (Structured):

- The function `getTheCandList` returns a structured list of all representatives.

```
function getTheCandList() public view returns (TheCand[] memory) {  infinite gas
    return representativesList;
}
```

12. Casting a Vote:

- The function `castBallot` allows a registered voter to cast a vote for a representative.

```
// Cast a vote
function castBallot(bytes32 _representativesHash) external {  infinite gas
    require(!TheVoters[msg.sender].hasVoted, "TheVoter voted");
    require(TheVoters[msg.sender].isRegistered, "TheVoters function");
    require(representativesExists(_representativesHash), "Not a valid representatives");

    allVotesIn++;
    updateTheCandVotes(_representativesHash);
    TheVoters[msg.sender].hasVoted = true;
}
```

13. Checking if Representative Exists:

- The internal function `representativesExists` checks whether a representative exists based on its hash.

```
// Check if a reps exists
function representativesExists(bytes32 _representativesHash) internal view returns (bool) {  infinite gas
    for (uint256 i = 0; i < representativesList.length; i++) {
        if (representativesList[i].hash == _representativesHash) {
            return true;
        }
    }
    return false;
}
```

14. Checking Votes for a Representative:

- The function `checkVotesPerGotRep` allows checking the vote count for a specific representative.

```
// Get the vote count for any one rep
function checkVotesPerGotRep(bytes32 _representativesHash) external view returns (uint256) {  infinite gas
    require(representativesExists(_representativesHash), "Not valid representatives");
    for (uint256 i = 0; i < representativesList.length; i++) {
        if (representativesList[i].hash == _representativesHash) {
            return representativesList[i].votes;
        }
    }
    return 0;
}
```


15. Updating Votes for a Representative:

- The internal function `updateTheCandVotes` updates the vote count for a specific representative.

```
// This is for updating the vote count for any one rep
function updateTheCandVotes(bytes32 _representativesHash) internal {  infinite gas
    for (uint256 i = 0; i < representativesList.length; i++) {
        if (representativesList[i].hash == _representativesHash) {
            representativesList[i].votes++;
            break;
        }
    }
}
```

16. Checking the Winner:

- The function `winnerChecking` checks and returns the winning representative based on the highest vote count.


```
//check who won
function winnerChecking() external view returns (TheCand memory winner) {  infinite gas
    require(representativesList.length > 0, "No representativess in memory");

    winner = representativesList[0];
    uint256 maxVotes = winner.votes;

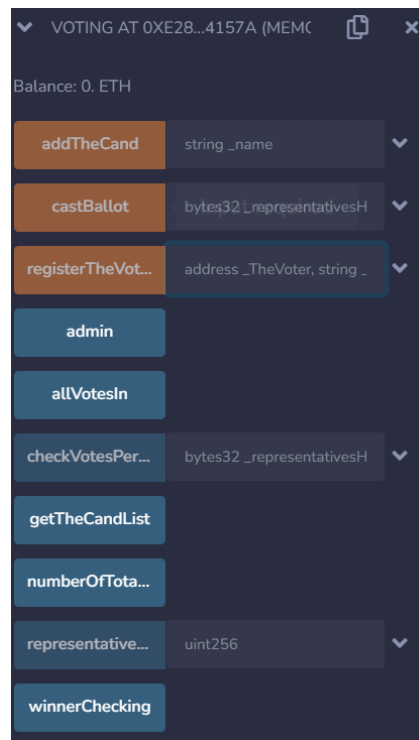
    for (uint256 i = 1; i < representativesList.length; i++) {
        if (representativesList[i].votes > maxVotes) {
            winner = representativesList[i];
            maxVotes = winner.votes;
        }
    }
}
```

17. Getting Total Number of Votes:

- The function `numberOfTotalVotes` returns the total number of votes cast.

```
// Get all recieved vote number
function numberOfTotalVotes() external view returns(uint256){  1205 gas
    return allVotesIn;
}
```

18. Program Deployment



19. Cryptographic methods used

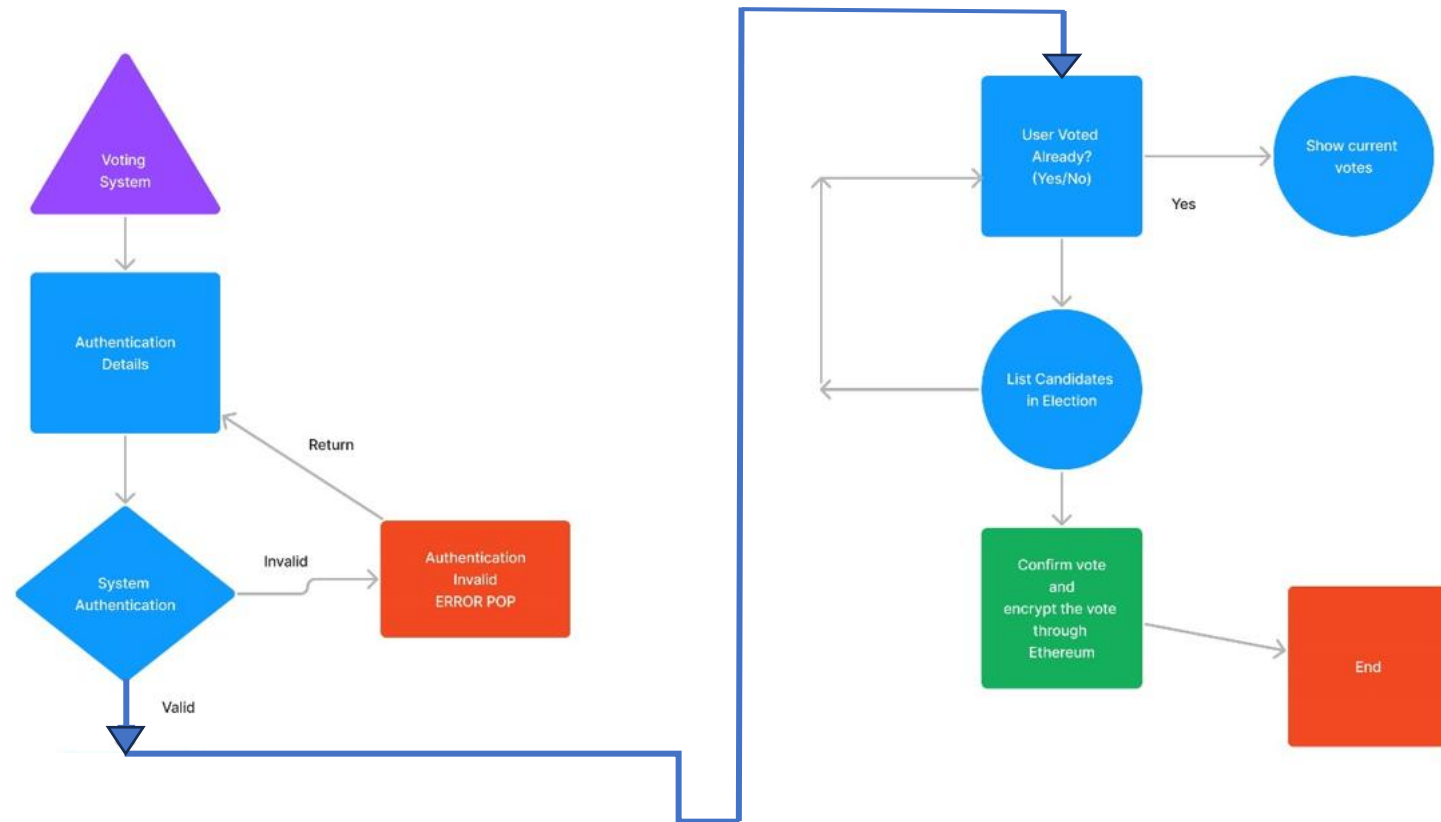
In our project we use a hash cryptography method.

Hashing:

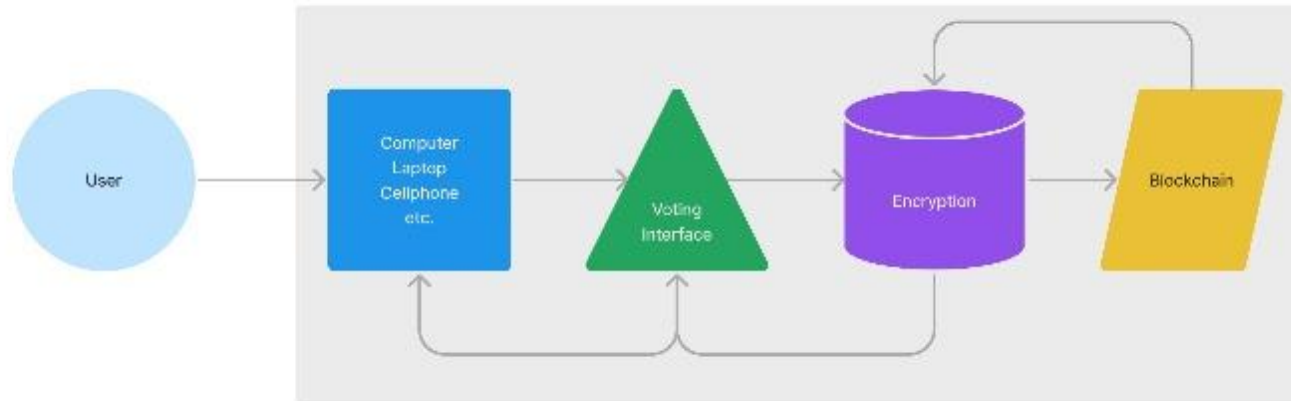
```
bytes32 hash = keccak256(bytes(_name)); // create a hash for representatives
```

- The code uses the `keccak256` function to generate a cryptographic hash (SHA-3) of a candidate's name. This hash is stored in the `hash` field of the `TheCand` struct.

4. Diagrams



Flowchart Blockchain Voting System



How would the blockchain voting system work.

5. Economic Advantages

5.1 Cost Analysis

- Internet and smartphone usage in countries must be over 90% for the proposed system, so the results can be trusted.
- Governments may need significant investments in infrastructure if not at this level.
- Increasing internet access worldwide supports the system's possibility.

The proposed voting system presents economic advantages, reducing labour and equipment costs in the long run. Initial costs for the first political race might be higher due to system implementation. However, subsequent election cycle costs are significantly reduced, offering a cost-effective solution for the entire country.

6. Conclusion

Our Blockchain voting system offers a solution to current challenges in traditional voting systems, introducing transparency, security, and cost-effectiveness. Leveraging Blockchain technology ensures the integrity of the election process, making it a promising solution for global electoral system modernization.

Our system, with its client-server architecture and smart contracts, addresses challenges in current voting systems such as the lack of voters' trust, makes it possible to audit the system, as well as making it more transparent. It aims to revolutionize elections by combining Blockchain's decentralization with smart contract security, ensuring voter privacy, preventing fraud, and maintaining overall electoral integrity.

Acknowledging potential challenges, including technical obstacles and the need for further research, the proposed system requires collaboration between technology experts, policymakers, and the public for successful implementation. There should be a heavy government incentive as well as public support for a change.

Moreover, integrating Blockchain and smart contracts in the electoral process means a significant step to improve worldwide democracy prospects. This ensures transparency, verifiability, and security in voting systems. Continuous advancements and testing of the system are essential for improving our proposed system and addressing possible concerns in the future.

Bibliography:

- Democracy thrived in ancient India* (2023) *Legacy IAS Academy*. Available at: <https://www.legacyias.com/democracy-thrived-in-ancient-india/> (Accessed: 29 November 2023).
- Desai, Z. and Lee, A. (2019) *Technology and protest: The political effects of electronic voting in India*. Available at: <https://static1.squarespace.com/static/587fef5417bffcdd148430d/t/5d4c4997c79fea000195851d/1565280664443/evm-paper---revision2.pdf> (Accessed: 30 November 2023).
- France 24 (2022) *Five things on Brazil's voting machines*, *France 24*. Available at: <https://www.france24.com/en/live-news/20220901-five-things-on-brazil-s-voting-machines> (Accessed: 30 November 2023).
- Gilberstadt, H. (2020) *Deep divisions in views of the election process – and whether it will be clear who won*, *Pew Research Center - U.S. Politics & Policy*. Available at: <https://www.pewresearch.org/politics/2020/10/14/deep-divisions-in-views-of-the-election-process-and-whether-it-will-be-clear-who-won/> (Accessed: 29 November 2023).
- Jafar, U., Aziz, M.J.A. and Shukur, Z. (2021) *Blockchain for Electronic Voting System-review and open research challenges*, *Sensors (Basel, Switzerland)*. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8434614/> (Accessed: 29 November 2023).
- What is an electronic voting machine and how it works?* (2023) *Digit Insurance*. Available at: <https://www.godigit.com/voter-id-card/what-is-an-electronic-voting-machine> (Accessed: 30 November 2023).

GitHub-Link:

<https://github.com/guifeliix/DDT-CA1>

Video-Link:

<https://drive.google.com/file/d/1tTf911qn2KNYnnyUUS0qXRxUjFn3w61S/view?usp=sharing>