ELECTRONIC VOTING SYSTEM

Team ID : NM2023TMID00666

TEAM LEADER: MUTHUVEL M - 951320106028

TEAM MEMBER 1: SAKTHIMATHIBALAN K - 951320106034

TEAM MEMBER 2: SARAVANA SELVAM S - 951320106037

TEAM MEMBER 3: ESAKKI MUTHU P - 951320106303

TABLE OF CONTENT

1. INTRODUCTION

1.1 Project Overview

The Blockchain-Based electronic voting system is a forward-looking technological solution designed to streamline and enhance various aspects of the real estate industry. By leveraging blockchain technology, this project aims to provide transparency, efficiency, and trust in property transactions, property management, and ownership.

1.2 Purpose

The purpose of an electronic voting system based on blockchain is to address various challenges and improve the overall voting process in democratic societies. Enhance the security of the voting process by using blockchain's tamper-proof and immutable ledger to record and store votes. Provide a transparent and auditable system where voters and relevant stakeholders can view the voting process, ensuring that the results are accurate and trustworthy. The open nature of the blockchain allows for public scrutiny.

2. LITERATURE SURVEY

2.1 Existing problem

Existing problems in electronic voting systems based on blockchain include scalability issues that can hamper their efficiency during large-scale elections, the complexity of the underlying technology, which may require significant technical expertise, and the challenge of educating voters on its proper use. Striking a balance between voter privacy and transparency poses a critical concern, while robust identity verification methods need to be developed to prevent fraudulent voting. Legal and regulatory hurdles can impede adoption, and cybersecurity risks demand constant vigilance to protect against cyberattacks. Ensuring inclusivity for those without access to the required technology and addressing the high implementation costs further complicate the adoption of these systems. Resistance to change and concerns about the security and fairness of electronic voting versus traditional methods remain significant obstacles. Lastly, reliance on internet connectivity and network reliability can affect the accessibility and trustworthiness of these systems. Addressing these problems is essential for the successful and widespread adoption of blockchain-based electronic voting.

2.2 References

Diponkar Paul and Suboj Kumar Ray, Member IACSIT, Vol. 3, No. 2, March 2013, “A preview n microcontroller Based electronic Voting machine”, International journal Of Information and Electronics Engineering.

D. Balzarotti, G. Banks, M. Cova, V. Felmetsger, R. A. Kemmerer, W. Robertson, F. Valeur, and G. Vigna, vol.36, No. 4, 2010. “An Experience in Testing the Security of Real-World Electronic Voting Systems”, IEEE Transactions on Software Engineering.

A. Villafiorita and K. Weldemariam, and R. Tiella, vol. 4, No. 4, 2009. “Development Formal Verification and Evaluation of an E-Voting System with VVPAT”, IEEE Transactions on Information Forensics and Security. http://www.bravenewballot.org/e-voting-in-india.html.

Anil K. Jain, Arun Ross and Salil Prabhakar, Vol. 14, No.1, January 2004. “An Introduction to Biometric Recognition”, IEEE Transactions on Circuits and Systems For Video Technology, Special Issue on Imageand Video Based Biometrics.

Anil K. Jain and Umut Uludag, Vol. 25, No. 11, pp.1094- 1098, Nov 2003. “Hiding Biometric Data”, IEEE Transactions on Pattern Analysis and Machine Intelligence. http://uidai.gov.in/aadhaar.html

S. Prabhakar, S. Pankanti, and A. K. Jain Vol. 1, No. 2, pp.33 -42, 2003 “Biometric Recognition: Security and Privacy Concerns”, IEEE Security and Privacy Magazine.

J. L. Wayman, Vol.1, No. 1, pp. 93-113, 2001, “Fundamentals of Biometric Authentication Technologies” International journal of Image and Graphics.

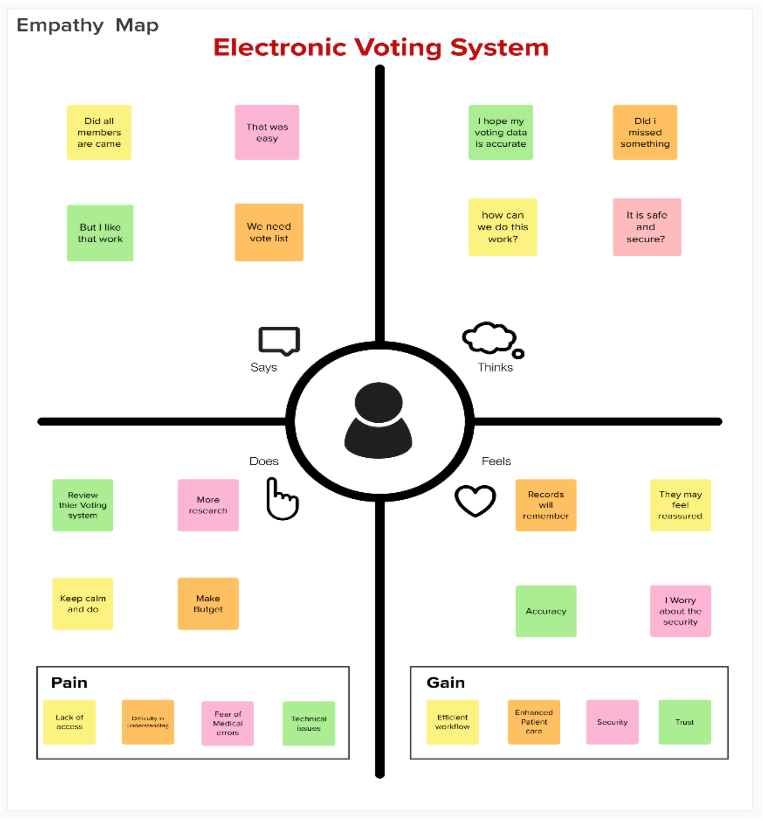
L. Hong, A. K. Jain, and S. Pankanti, ProcAutoID'99s, Pp.59-64, Oct 1999 “Can Multi Biometrics Improve Performance”? Summit (NJ), USA..

2.3 Problem Statement Definition

An electronic voting system based on blockchain is a digital platform designed to facilitate secure and transparent voting processes. It leverages blockchain technology to record and store votes in a decentralized and immutable manner. Each vote is encrypted, time-stamped, and added to a chain of blocks, making it extremely difficult to tamper with or manipulate the results. This technology aims to enhance the integrity of elections by providing transparency, security, and trust in the voting process.

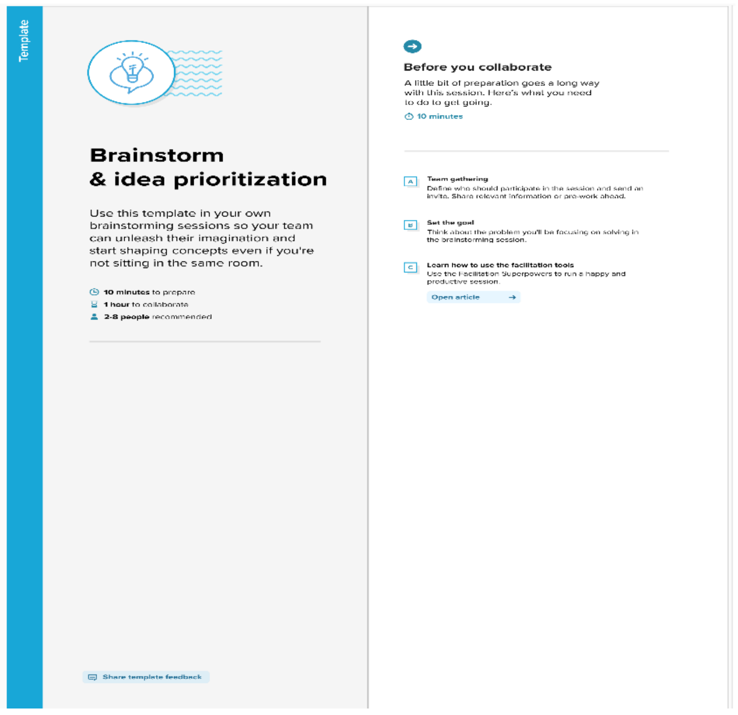
3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstormin

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Template

Brainstorm
& idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Share template feedback

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

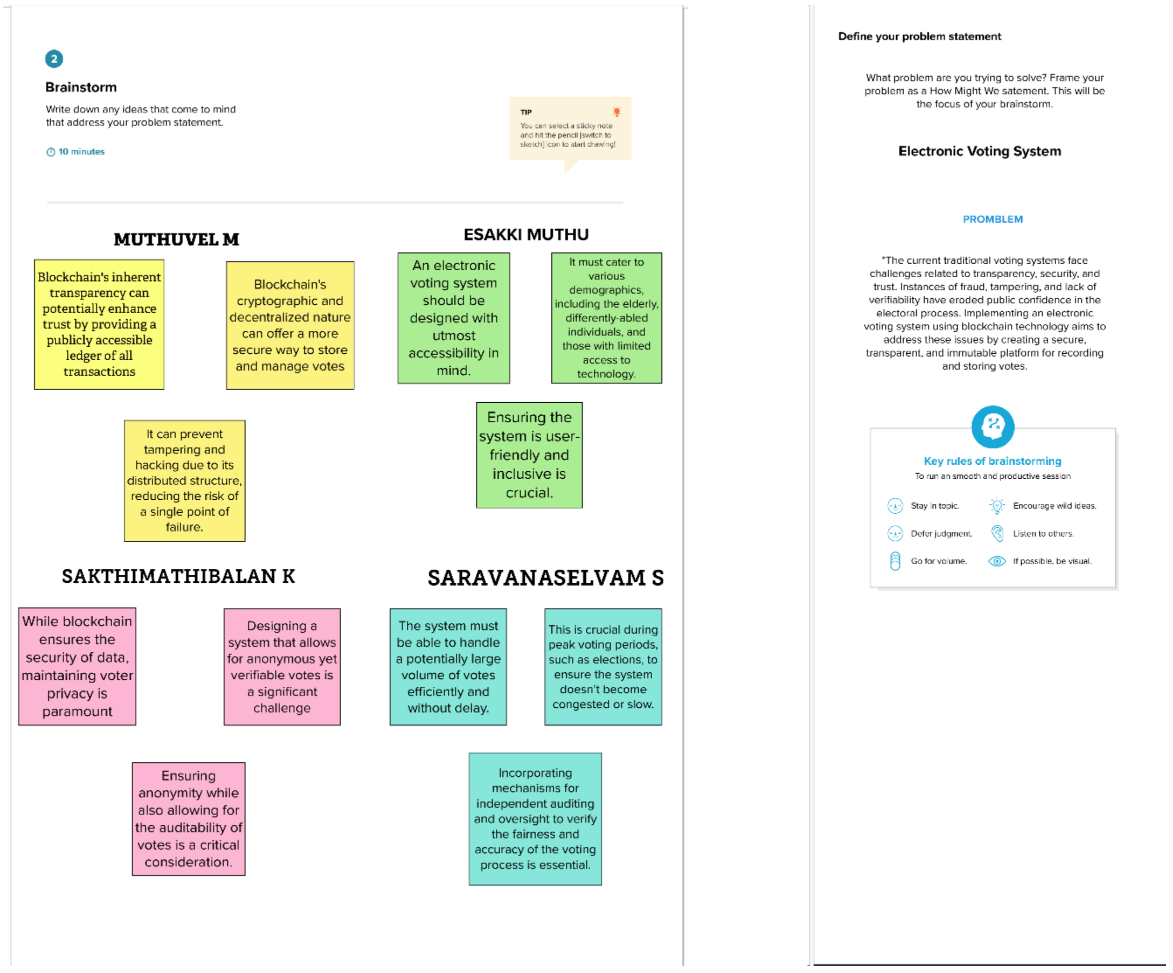
A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

Step-2: Brainstorm, Idea Listing and Grouping



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

MUTHUVEL M

Blockchain's inherent transparency can potentially enhance trust by providing a publicly accessible ledger of all transactions

Blockchain's cryptographic and decentralized nature can offer a more secure way to store and manage votes

It can prevent tampering and hacking due to its distributed structure, reducing the risk of a single point of failure.

ESAKKI MUTHU

An electronic voting system should be designed with utmost accessibility in mind.

It must cater to various demographics, including the elderly, differently-abled individuals, and those with limited access to technology.

Ensuring the system is user-friendly and inclusive is crucial.

SAKTHIMATHIBALAN K

While blockchain ensures the security of data, maintaining voter privacy is paramount

Designing a system that allows for anonymous yet verifiable votes is a significant challenge

Ensuring anonymity while also allowing for the auditability of votes is a critical consideration.

SARAVANASELVAM S

The system must be able to handle a potentially large volume of votes efficiently and without delay.

This is crucial during peak voting periods, such as elections, to ensure the system doesn't become congested or slow.

Incorporating mechanisms for independent auditing and oversight to verify the fairness and accuracy of the voting process is essential.

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We satement. This will be the focus of your brainstorm.

Electronic Voting System

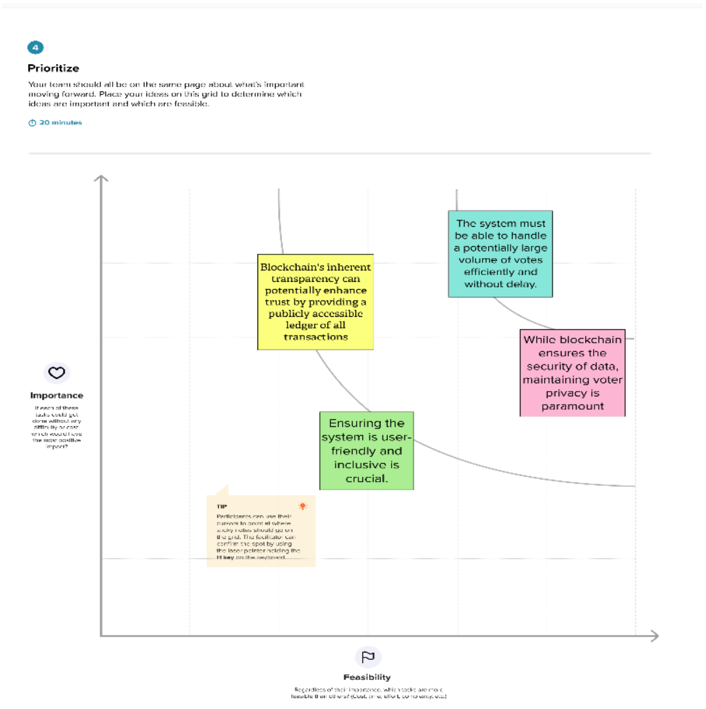
PROMBLEM

"The current traditional voting systems face challenges related to transparency, security, and trust. Instances of fraud, tampering, and lack of verifiability have eroded public confidence in the electoral process. Implementing an electronic voting system using blockchain technology aims to address these issues by creating a secure, transparent, and immutable platform for recording and storing votes.

Key rules of brainstorming
To run an smooth and productive session

Stay in topic.
Encourage wild ideas.
Defer judgment.
Listen to others.
Go for volume.
If possible, be visual.

Step-3: Idea Prioritization


4
Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
20 minutes
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Blockchain's inherent transparency can potentially enhance trust by providing a publicly accessible ledger of all transactions
The system must be able to handle a potentially large volume of votes efficiently and without delay.
While blockchain ensures the security of data, maintaining voter privacy is paramount
Ensuring the system is user-friendly and inclusive is crucial.
TIP
Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

1.Vs Code

Visual Studio Code (VS Code) is a versatile and popular code editor that can be used for developing blockchain-based electronic voting system. Here are some steps and tips for using VS Code in such a project:

Installation:

Download and install Visual Studio Code from the official website (https://code.visualstudio.com/).

Extensions:

Install relevant extensions for blockchain and contract development. Some commonly used extensions for Ethereum and Solidity development include "Solidity" and "Truffle."

Contract Development:

- Create a new folder for your project and open it in VS Code.
- Write, test, and deploy your contracts using the Solidity extension.
- Use Truffle, a development framework for Ethereum, to help you with contract development.

Front-End Development:

Develop the front-end of your electronic voting system using web development technologies like HTML, CSS, and JavaScript. Ensure that your front-end code interacts with your   contracts on the blockchain.

Testing:

Write tests for your   contracts and front-end code. VS Code can be used to run and manage test suites.

Deployment:

Deploy your   contracts and front-end applications to the desired blockchain network and hosting platforms.

2. Nodejs

Node.js is a popular runtime environment for JavaScript, and it can be used in the development of a Blockchain-Based electronic voting system, particularly for building the backend server, handling API requests, and interacting with the blockchain. Here's how Node.js can be integrated into the development process:

Backend Development:

Node.js is commonly used to build the backend server of web applications. You can create an Express.js server to handle API requests from the front end and interact with the blockchain network (e.g., Ethereum).

Contract Deployment:

You can use Node.js scripts to automate the deployment of   contracts to the blockchain network during the development and testing phases.

Authentication and Authorization:

Implement user authentication and authorization using Node.js middleware, such as Passport.js, to secure **your application and control access to certain features.**

Real-Time Features:

For real-time features like property bidding, chat, or notifications, you can use Node.js with WebSocket libraries like Socket.io to provide instant updates to users.

4.2 Non-Functional requirements

1. Metamask

MetaMask is a popular cryptocurrency wallet and decentralized application (dApp) browser extension that allows users to interact with the Ethereum blockchain. It serves as both a cryptocurrency wallet and a gateway to the world of decentralized applications.

MetaMask has gained popularity for its role in enabling users to access and participate in the decentralized finance (DeFi) ecosystem, interact with non-fungible tokens (NFTs), and securely manage their Ethereum-based assets

Integrating MetaMask into a Blockchain-Based electronic voting system can offer several benefits, especially when dealing with Ethereum-based tokens, contracts, and decentralized applications (dApps).

User Wallets: Each user, including property owners, buyers, and investors, can have their own MetaMask wallet. This wallet allows them to securely store and manage property tokens, Ether (ETH), or any other cryptocurrencies used within the system.

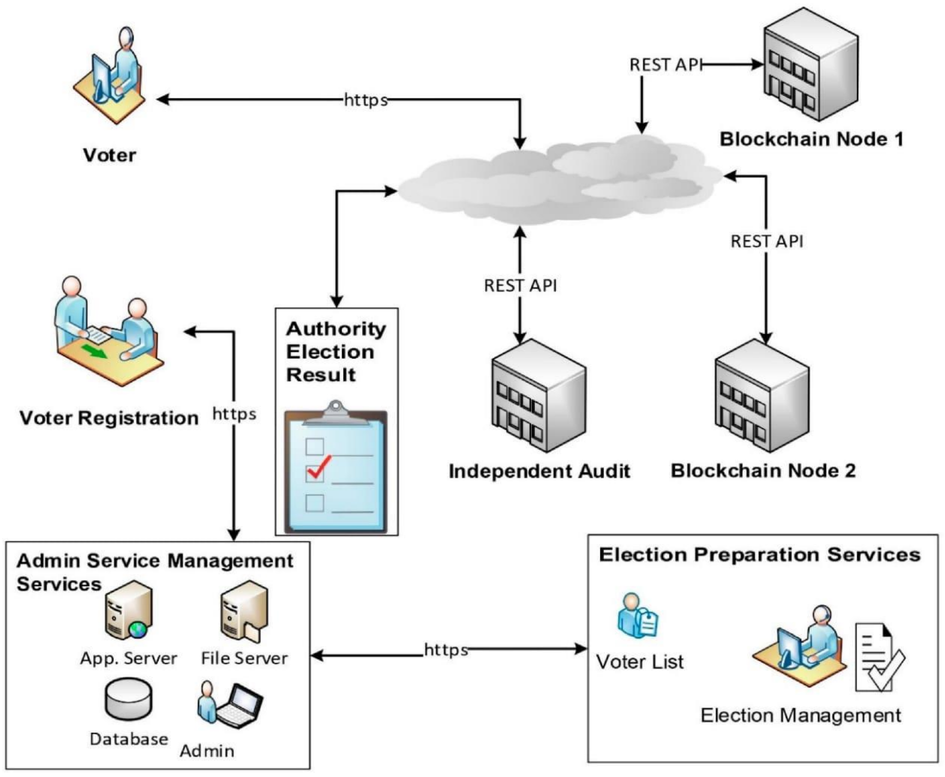
Access to dApps: MetaMask serves as a bridge to decentralized applications. Users can access the electronic voting system's dApp through MetaMask, enabling them to initiate property transactions, view property details, and perform various management tasks.

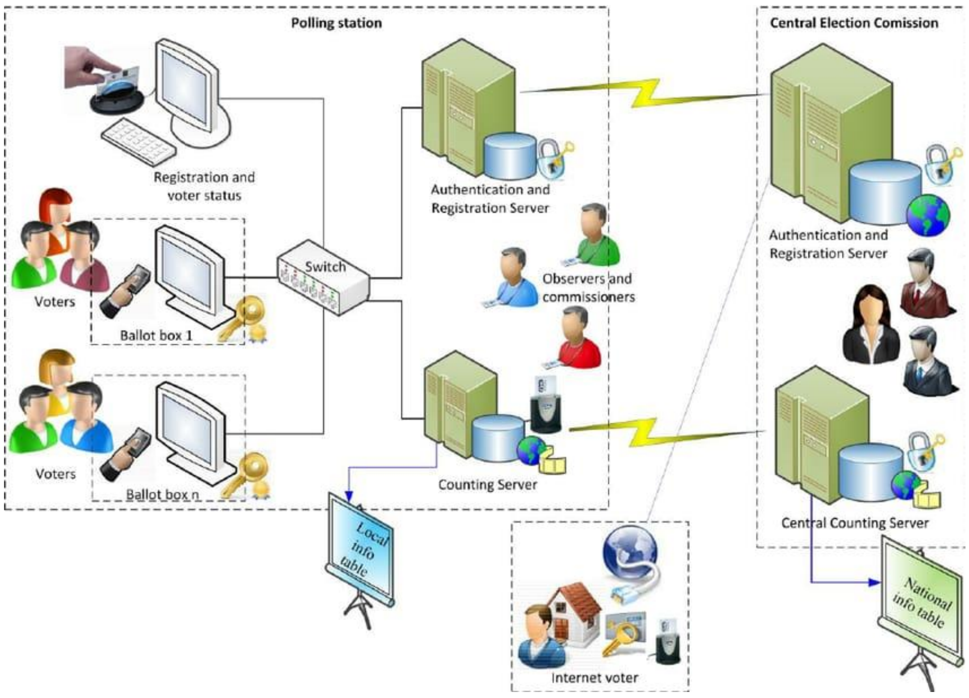
Transaction Execution: Users can use MetaMask to interact with the contracts that govern property transactions. For example, they can initiate property purchases, sales, and transfers directly through their MetaMask wallets.

Token Management: Property tokens, which represent fractional ownership, can be managed within MetaMask. Users can view their token holdings, transfer tokens to other users, and participate in property investment activities.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User & Stories Solution Architecture




Polling station
Central Election Comission
Registration and voter status
Authentication and Registration Server
Voters
Switch
Ballot box 1
Observers and commissioners
Authentication and Registration Server
Voters
Ballot box n
Counting Server
Central Counting Server
Local info table
Internet voter
National info table

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

The technical architecture of an electronic voting system based on blockchain typically involves several key components and layers. Below is a simplified overview of the technical architecture:

Voter Interface: This is the user-facing part of the system where voters interact to cast their votes. It can be a web application, a mobile app, or even specialized voting machines.

Identity Verification: Ensuring the identity of voters is a critical step. This can involve methods like biometric authentication, government-issued IDs, or digital signatures.

Vote Casting: Once identity is verified, voters cast their votes through the voter interface. Votes are encrypted to ensure privacy.

Blockchain Network: The blockchain network itself consists of nodes (computers) that maintain the distributed ledger. The blockchain records all votes in a transparent and immutable manner. There are two main types of blockchain networks used:

a. Public Blockchain: Offers maximum transparency and security, with open participation from anyone. Examples include Ethereum or Bitcoin.

b. Private Blockchain: Provides more control and privacy, often used by organizations or governments. Hyperledger Fabric is a popular choice.

Smart Contracts: These are self-executing contracts with the terms of the agreement directly written into code. In a voting system, smart contracts handle vote validation, counting, and result determination.

Consensus Mechanism: Consensus algorithms (e.g., Proof of Work, Proof of Stake) are used to validate and agree on the state of the blockchain, ensuring that only valid votes are recorded.

Security Layer: This includes cryptographic techniques for securing votes, access control, and encryption.

Storage and Data Access Layer: The recorded votes and smart contract data are stored securely. Access to this data may be restricted to authorized parties.

User Verification and Registration: This component handles the voter registration process, ensuring that eligible voters can participate.

Monitoring and Auditing: Real-time monitoring and auditing tools can be implemented to oversee the integrity of the election process and the blockchain network.

Results Reporting: After voting concludes, the results are computed and reported through the blockchain. These results can be accessed by authorized parties and the public.

External Systems Integration: Depending on the requirements, the system may need to integrate with other systems such as government databases for voter registration or notification services.

User Support and Helpdesk: A system for assisting voters who may face issues during the voting process.

It's essential to note that the exact architecture can vary based on the specific requirements of a given election, regulatory constraints, and the level of decentralization desired. Security and privacy are of utmost concern in designing such systems to ensure the integrity of the voting process.

DEFINE PROJECT OBJECTIVES AND SCOPE:

Clearly define the goals and scope of the project.

1. Determine the type of elections the system will support (national, local, organizational).

2. Stakeholder Analysis:

Identify and engage with all relevant stakeholders, including government agencies, election commissions, voters, and technology partners.

3. Legal and Regulatory Compliance:

Ensure the project complies with all relevant laws and regulations related to elections, data privacy, and blockchain technology.

4. Requirements Gathering:

Define detailed requirements for the electronic voting system. This includes functional, technical, and security requirements.

5. System Design:

Create a detailed system design that outlines the architecture, technology stack, and components of the electronic voting system. Consider scalability, security, and usability.

6. Technology Selection:

Choose the appropriate blockchain platform (public or private), consensus mechanism, and any other necessary technologies (smart contracts, encryption, identity verification).

7. Risk Assessment:

Identify potential risks related to the project and develop a risk mitigation plan.

8. Timeline and Milestone Planning:

Create a project timeline with well-defined milestones. These milestones might include system development, security audits, and testing phases.

9. Resource Allocation:

Assign human and financial resources to the project. This includes hiring or training staff with the required expertise.

10. Development and Testing:

Begin the development of the electronic voting system, including smart contracts and user interfaces. Extensive testing, including security testing, should be an integral part of this phase.

11. Voter Registration:

Implement the voter registration process, ensuring that eligible voters are properly verified and registered.

12. Security Measures:

Implement robust security measures to protect against hacking and fraud, including encryption, authentication, and secure access controls.

13. Training and Education:

Train election officials, staff, and voters on how to use the system. Educational materials and support channels should be established.

14. Pilot Testing:

Conduct a small-scale pilot test of the electronic voting system to identify and resolve any issues before the full-scale deployment.

15. Full-Scale Deployment:

Roll out the system for use in actual elections, while closely monitoring its performance.

16. Monitoring and Auditing:

Implement real-time monitoring and auditing tools to oversee the integrity of the election process and the blockchain network.

17. Results Reporting:

Develop a process for reporting election results transparently and efficiently.

18. Evaluation and Continuous Improvement:

After the elections, evaluate the system's performance and gather feedback from stakeholders for continuous improvement.

19. Post-Deployment Support:

Provide ongoing support and maintenance for the electronic voting system.

20. Documentation and Reporting:

Maintain comprehensive documentation of the project, including techical specification, audit reports and compliance records.

21. Public Communication:

Maintain open and transparent communication with the public and stakeholders regarding the system's security, reliability, and performance.

22. Contingency Plans:

Develop contingency plans in case of unforeseen issues, such as cyberattacks or technical failures.

23. Project Closure:

Close the project, including final reporting, documentation, and archiving of project records.

Project planning and scheduling for an electronic voting system based on blockchain is a complex and critical process. It requires a multidisciplinary team with expertise in blockchain technology, cybersecurity, election administration, and legal compliance. Additionally, close collaboration with government authorities and election commissions is essential to ensure a smooth implementation.

6.2 Sprint Planning & Estimation

Sprint planning and estimation are crucial components of an agile development process for a Blockchain-Based electronic voting system. They help the development team organize work, set priorities, and estimate the effort required for each task within a sprint. Here's how sprint planning and estimation can be approached:

1. Product Backlog:

Begin with a well-defined product backlog. This backlog includes all the features, user stories, and tasks that need to be implemented in the system. It should be maintained and prioritized by the product owner.

2. Sprint Goal:

Determine the sprint goal for the upcoming sprint. This goal should be aligned with the overall project objectives and should specify what the team intends to achieve in the sprint.

3. Sprint Length:

Decide on the duration of the sprint. Common sprint lengths are 2 weeks, 3 weeks, or 4 weeks. Choose a sprint length that suits the team's work capacity and project complexity.

4. **Sprint Planning Meeting:**

Conduct a sprint planning meeting before the start of each sprint. This meeting involves the product owner, development team, and the scrum master. During the meeting, the team reviews the prioritized backlog items and selects the ones to be worked on in the upcoming sprint

Sprint planning and estimation are iterative processes, and the team should continuously improve their estimation accuracy and sprint planning based on past performance and feedback. Effective sprint planning and estimation ensure that the Blockchain-Based electronic voting system is developed efficiently and with a focus on delivering value to users.

6.3 Sprint Delivery Schedule

The sprint delivery schedule for a Blockchain-Based electronic voting system depends on the sprint length and the scope of work for each sprint. Typically, agile development teams follow a sprint schedule with fixed sprint durations (e.g., 2 weeks, 3 weeks, or 4 weeks). Here's an example of a sprint delivery schedule for a 2-week sprint:

Sprint 1:

Sprint Duration: Week 1 and Week 2

Sprint Planning: Day 1 of Week 1

Daily Standup Meetings: Held daily throughout the sprint

Mid-Sprint Review: Day 5 of Week 1

Sprint Review and Demo: Day 5 of Week 2

Sprint Retrospective: Day 5 of Week 2

7. CODING & SOLUTIONING

```
 // SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;



contract VoteSystem{
   address public owner;



   constructor(){
      owner= msg.sender;
   }



 struct candidate {
    uint voterId;
    string name;
    uint age;
    uint voteCount;
 }

 mapping (uint => candidate) candidateMap;

 struct voters {
    uint voterId;
    string name;
    uint age;
    bool votingState;
 }


 mapping (uint => voters) votersMap;
 mapping (uint=>bool) registeredVoter;

 modifier checkVoterVoted(uint _votersVoterId){
    require (votersMap[_votersVoterId].votingState == false);
    _;
 }


modifier checkRegisteredVoter(uint _votersVoterId){
      require(registeredVoter[_votersVoterId]==true, "Voter is not Registered");
      _;
}
uint[] voterIdlist;
uint[] candidateIdList;
```

```
 }

 function enrollCandidate(uint _voterId,string memory _name,uint  _age )  public {


 require (_age >= 25);
 require (candidateMap[_voterId].voterId != _voterId);


   candidateMap[_voterId].voterId = _voterId;
   candidateMap[_voterId].name = _name;
   candidateMap[_voterId].age = _age;

   candidateIdList.push(_voterId);
 }


 function enrollVoter(uint _voterId,string memory _name,uint _age)  public  returns(bool){


require (_age >= 18);
require (votersMap[_voterId].voterId != _voterId);


    votersMap[_voterId].voterId = _voterId;
    votersMap[_voterId].name = _name;
    votersMap[_voterId].age = _age;

    voterIdlist.push(_voterId);
   return registeredVoter[_voterId]=true;


 }

 function getCandidateDetails(uint _voterId) view public returns(uint,string memory,uint,uint) {
    return
(candidateMap[_voterId].voterId,candidateMap[_voterId].name,candidateMap[_voterId].age,cand
idateMap[_voterId].voteCount);

 }


 function getVoterDetails(uint _voterId) view public returns (uint,string memory,uint,bool){


    return
(votersMap[_voterId].voterId,votersMap[_voterId].name,votersMap[_voterId].age,votersMap[_vo
terId].votingState);


 }


 function vote(uint _candidateVoterId,uint _votersVoterId) public checkVoterVoted(_votersVoterId)
checkRegisteredVoter(_votersVoterId) {

    candidateMap[_candidateVoterId].voteCount += 1;

    votersMap[_votersVoterId].votingState = true;

 }


 function getVotecountOf(uint _voterId) view public returns(uint){

      require(msg.sender== owner, "Only owner is allowed to Check Results");

    return candidateMap[_voterId].voteCount;

 }


 function getVoterList() view public returns (uint[] memory){



   return   voterIdlist;

   }


 function getCandidateList() view public returns(uint[] memory){
```

```
 return candidateIdList;
 }
```

8. PERFORMANCE TESTING

1. Define Performance Metrics:

Determine the key performance metrics to be measured, such as transaction processing speed, system response time, throughput, and resource utilization (CPU, memory, network).

2. Test Environment Setup:

Create a testing environment that closely mimics the production environment, including the same hardware, network conditions, and configurations.

3. Load Testing:

Simulate the expected load on the system by creating a large number of virtual users (representing voters) and submit votes concurrently. Observe how the system handles this load.

4. Scalability Testing:

Determine the system's ability to scale horizontally (adding more nodes or servers) or vertically (upgrading server resources) to accommodate increased load.

5. Stress Testing:

Push the system beyond its expected limits to identify breaking points and measure system stability under extreme conditions.

6. Security Testing:

Assess the system's ability to withstand various security threats, including DDoS attacks and attempts to compromise the integrity of the blockchain.

7. Usability Testing:

Evaluate the system's ease of use and responsiveness from the voter's perspective. Ensure that the user interface is intuitive and efficient.

8. Network Latency Testing:

Assess how the system performs under different network conditions, including high-latency connections, to ensure that remote voters can participate effectively.

9. Fault Tolerance Testing:

Introduce failures in the system, such as node crashes or network interruptions, to validate that the blockchain network can recover and continue functioning.

10. Performance Optimization:

Identify bottlenecks and areas for improvement. Optimize the system's components, including smart contracts, consensus mechanisms, and database queries.

11. Data Management Testing:

Evaluate the efficiency of data storage and retrieval within the blockchain, ensuring that historical voting data can be managed effectively.

12. Response Time Measurement:

Measure the time it takes for a vote to be recorded in the blockchain and for voters to receive confirmation. Ensure that response times are within acceptable limits.

13. Monitoring and Reporting:

Implement real-time monitoring tools to capture performance data and generate reports. Set up alerts for abnormal performance conditions.

14. Failover and Recovery Testing:

Test the system's ability to recover from unexpected failures and ensure that backup mechanisms are functioning correctly.

15. Data Integrity Testing:

Verify that all votes recorded in the blockchain are accurate and haven't been tampered with. Ensure the immutability of the blockchain.

16. Load Balancing Testing:

Assess how the system distributes loads across multiple nodes or servers to maintain optimal performance.

17. Endurance Testing:

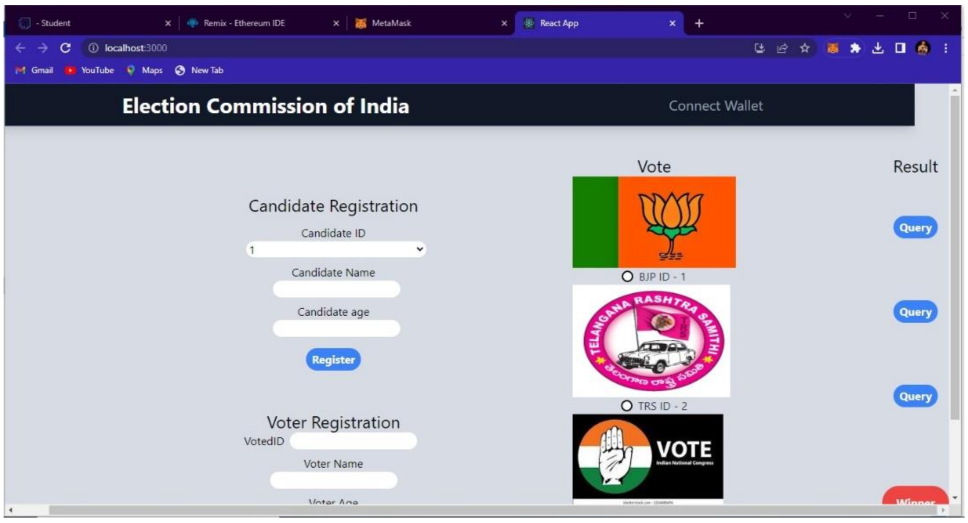
Run the system under a sustained load for an extended period to identify issues related to resource exhaustion, memory leaks, and system degradation over time.

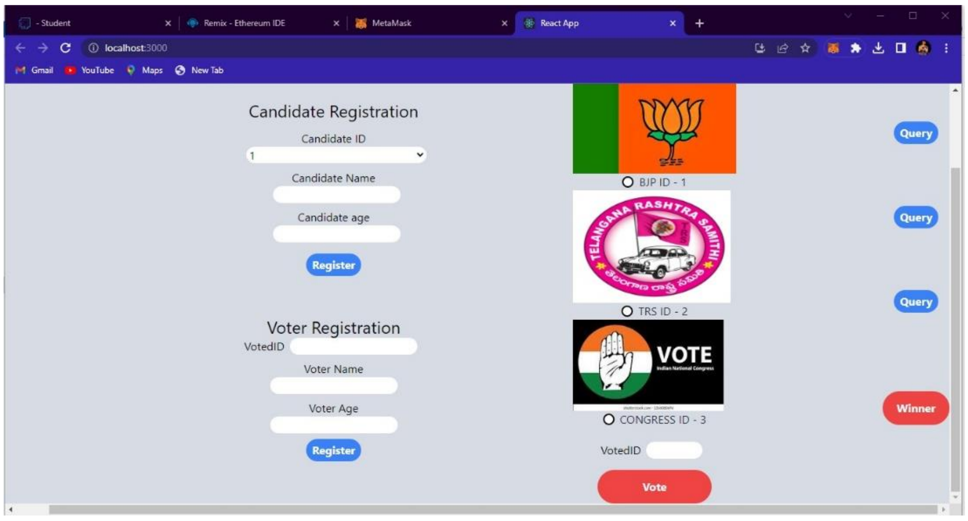
18. Compliance Testing:

Ensure that the system complies with legal and regulatory requirements for electronic voting, data protection, and privacy.

9. RESULTS

9.1 Output Screenshots





10. ADVANTAGES & DISADVANTAGES

Advantages:

The advantages of an electronic voting system based on blockchain include:

Transparency: Blockchain technology provides a transparent and publicly accessible ledger of all votes cast, making it difficult to manipulate or alter the results without detection.

Security: The use of cryptographic techniques and decentralization in blockchain ensures a high level of security, reducing the risk of hacking or fraud in the voting process.

Trust: Blockchain's immutability and tamper-resistant features build trust in the electoral system, as voters can verify their votes and election results independently.

Accessibility: Electronic voting systems can be designed to improve accessibility for all voters, including those with disabilities, by offering various voting methods and easy-to-use interfaces.

Efficiency: Electronic systems can streamline the voting process, reducing the time required for counting and announcing results, which is particularly beneficial for large-scale elections.

Reduced Costs: Over time, electronic voting systems based on blockchain can potentially reduce the costs associated with paper-based elections, including printing, transportation, and manual counting.

Increased Voter Participation: Online voting options can make it more convenient for people to participate in elections, potentially increasing voter turnout.

Elimination of Intermediaries: Blockchain can reduce the need for intermediaries in the voting process, enhancing the security and integrity of the system.

Auditability: Blockchain's transparency allows for easy auditing of the entire voting process, ensuring that the results align with the votes cast.

Resistance to Single Points of Failure: The decentralized nature of blockchain makes it resistant to single points of failure, making it harder for malicious actors to disrupt the voting process.

These advantages collectively contribute to a more trustworthy, efficient, and accessible electoral system when blockchain technology is applied to electronic voting. However, it's essential to address challenges and concerns, such as voter privacy and the digital divide, to fully harness these benefits.

Disadvantages:

Technology Barriers: Not all voters may be comfortable or familiar with the technology required to participate in electronic voting, potentially disenfranchising some individuals, especially older or less tech-savvy voters.

Security Concerns: While blockchain is generally secure, no system is entirely immune to cyberattacks. Malicious actors may attempt to exploit vulnerabilities in the electronic voting system, compromising the integrity of the election.

Privacy Issues: Maintaining voter privacy while using blockchain can be challenging. Balancing transparency with the secrecy of the ballot is essential to prevent vote-buying or coercion.

Identity Verification: Ensuring that voters are who they claim to be in an online environment is a significant challenge. Secure and reliable identity verification mechanisms are necessary to prevent voter fraud.

Digital Divide: Electronic voting assumes widespread internet access and digital literacy. The digital divide may exclude individuals with limited access to the internet or those who are uncomfortable with digital technology.

System Complexity: Implementing and maintaining an electronic voting system based on blockchain requires expertise, and errors in the system's design or implementation can lead to problems or vulnerabilities.

Paper Backup: To ensure the integrity of the process, some argue that electronic voting systems should have paper backups to verify results, which adds a layer of complexity.

Legal and Regulatory Challenges: Developing and enforcing legal frameworks and regulations for electronic voting systems can be complex and vary from one jurisdiction to another.

Infrastructure Requirements: Building the necessary infrastructure for electronic voting systems can be costly and time-consuming, especially in regions with limited resources.

Resistance to Change: Some stakeholders, including government officials and citizens, may be resistant to adopting new voting technologies due to concerns about reliability, security, and the potential for manipulation.

Lack of Consensus: There is no universal agreement on the best approach to electronic voting using blockchain technology, leading to a lack of standardized systems and practices.

While electronic voting systems based on blockchain offer many benefits, these disadvantages and challenges must be carefully considered and addressed to ensure the successful and secure implementation of such systems in real-world elections.

11. CONCLUSION

The Electronic Voting System based on Blockchain presents a promising solution to address the challenges of traditional voting systems. By leveraging blockchain technology, it aims to create a more secure, transparent, and accessible electoral process that can enhance trust and confidence in democratic procedures.

12. FUTURE SCOPE

Enhanced Security: Blockchain's decentralized and cryptographic nature makes it inherently secure. It can help protect against tampering and hacking, increasing trust in the electoral process.

Transparency: The transparent and immutable nature of blockchain provides a clear audit trail for all votes cast. This transparency can significantly reduce concerns about fraud and manipulation.

Reduced Costs: Over time, electronic voting systems can be more cost-effective than traditional paper-based systems. Savings can be achieved through reduced printing, transportation, and manual counting expenses.

Increased Voter Participation: E-voting can make it more convenient for people to participate in elections, potentially leading to higher voter turnout. This could result in more representative and democratic outcomes.

Accessibility: Electronic voting systems can be designed to accommodate a wide range of voters, including those with disabilities. This promotes inclusivity in the electoral process.

Speed and Efficiency: Electronic voting can streamline the voting process, reducing the time required for counting and result reporting. Faster results can improve the overall efficiency of elections.

Eradicating Voter Fraud: Blockchain's security features can help eliminate voter fraud by ensuring that each vote is counted only once and by providing a transparent audit trail.

Remote Voting: Electronic voting systems can enable remote voting, allowing citizens to vote from anywhere, which can be especially beneficial in situations like pandemics when in-person voting may be risky.

Resilience to Cyberattacks: A well-designed blockchain-based system can be resistant to cyberattacks due to its decentralized nature, making it a robust option for protecting the integrity of elections.

Global Adoption: As blockchain-based voting systems gain acceptance and trust, they have the potential to become a standard for secure and transparent elections worldwide.

It's important to note that while blockchain-based electronic voting systems offer significant hope for improving the electoral process, they also come with challenges that need to be addressed, such as privacy concerns, technology barriers, and legal frameworks. Successful implementation requires careful planning and consideration of these challenges.

13. APPENDIX

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;


contract VoteSystem{
   address public owner;


   constructor(){
      owner= msg.sender;
   }


 struct candidate {
    uint voterId;
    string name;
    uint age;
    uint voteCount;
 }

 mapping (uint => candidate) candidateMap;

 struct voters {
    uint voterId;
    string name;
    uint age;
    bool votingState;
 }


 mapping (uint => voters) votersMap;
 mapping (uint=>bool) registeredVoter;

 modifier checkVoterVoted(uint _votersVoterId){
    require (votersMap[_votersVoterId].votingState == false);
    _;
 }


modifier checkRegisteredVoter(uint _votersVoterId){
       require(registeredVoter[_votersVoterId]==true, "Voter is not Registered");
       _;
}
uint[] voterIdlist;
uint[] candidateIdList;




 function enrollCandidate(uint _voterId,string memory _name,uint  _age )  public {


 require (_age >= 25);
 require (candidateMap[_voterId].voterId != _voterId);


   candidateMap[_voterId].voterId = _voterId;
   candidateMap[_voterId].name = _name;
   candidateMap[_voterId].age = _age;

   candidateIdList.push(_voterId);
 }

 function enrollVoter(uint _voterId,string memory _name,uint _age)  public  returns(bool){

require (_age >= 18);
require (votersMap[_voterId].voterId != _voterId);

    votersMap[_voterId].voterId = _voterId;
    votersMap[_voterId].name = _name;
    votersMap[_voterId].age = _age;

    voterIdlist.push(_voterId);
   return registeredVoter[_voterId]=true;

 }

 function getCandidateDetails(uint _voterId) view public returns(uint,string memory,uint,uint) {

    return
(candidateMap[_voterId].voterId,candidateMap[_voterId].name,candidateMap[_voterId].age,cand
idateMap[_voterId].voteCount);
 }

 function getVoterDetails(uint _voterId) view public returns (uint,string memory,uint,bool){
    return
(votersMap[_voterId].voterId,votersMap[_voterId].name,votersMap[_voterId].age,votersMap[_vo
terId].votingState);


 }


 function vote(uint _candidateVoterId,uint _votersVoterId) public
checkVoterVoted(_votersVoterId) checkRegisteredVoter(_votersVoterId) {
    candidateMap[_candidateVoterId].voteCount += 1;
    votersMap[_votersVoterId].votingState = true;
 }

 function getVotecountOf(uint _voterId) view public returns(uint){
       require(msg.sender== owner, "Only owner is allowed to Check Results");
    return candidateMap[_voterId].voteCount;
 }

 function getVoterList() view public returns (uint[] memory){


    return   voterIdlist;
    }

 function getCandidateList() view public returns(uint[] memory){



 return candidateIdList;
 }
```

GitHub & Project Demo Link

Github link : https://github.com/Muthuvel-2003/NM2023TMID00666

Project Demo link : https://youtu.be/hl2SXVB6-_w