PROJECT REPORT

ELECTRONIC VOTING MACHINE

SUBMITTED BY:

SUBHIKSHA.V

SUBITHA.S

SIVASREE.S.S

SANTHANAMARI

TABLE OF CONTENTS

INTRODUCTION:

PROJECT OVERVIEW:

The integration of electronic voting machines (EVMs) with block chain technology is poised to revolutionize the way elections are conducted, ensuring transparency, security, and trust in the electoral process. EVMs have streamlined voting procedures, but they have faced security concerns. Block chain technology addresses these issues by providing a secure, transparent, and immutable platform for recording and verifying votes. This combination offers a promising solution to enhance the integrity of elections and boost public confidence in the democratic process.

PURPOSE:

Enhance Security: Block chain's cryptographic and decentralized nature makes it extremely difficult for malicious actors to tamper with or manipulate voting data, ensuring the security and integrity of the electoral process.

Ensure Transparency: By providing a transparent, publicly accessible ledger of votes, block chain technology allows all stakeholders to verify the authenticity of the voting data, thereby increasing trust in the electoral system.

Prevent Tampering: Once recorded on the block chain, votes become immutable and tamper-proof, safeguarding the integrity of the election's historical data and ensuring that no one can alter or delete votes.

Provide Traceability: Block chain enables voters to verify that their votes have been accurately recorded and counted, promoting accountability and trust in the voting system.

Improve Efficiency: Electronic voting with block chain can reduce the time and resources required for vote counting and result dissemination, expediting the election process and delivering timely results.

Mitigate Fraud: By leveraging block chain's consensus mechanisms, EVMs can help prevent fraud and ensure that only valid votes are counted, thus preserving the integrity of the electoral outcome.

Empower Remote Voting: Block chain-based EVMs can facilitate secure remote voting, making it more convenient for citizens to participate in the electoral process, including those who may not have easy access to physical voting locations.

LITERATURE SURVEY:

EXISTING PROBLEM:

Accessibility and Inclusivity: Ensuring that electronic voting is accessible to all citizens, regardless of their technological access or abilities.

Security Concerns: Addressing potential cyber threats, software vulnerabilities, and large-scale block chain breaches.

Identity Verification: Developing a reliable method for securely verifying voter identities online.

Privacy: Balancing voter privacy with block chain's transparency and auditability.

Block chain Scalability: Handling a large number of votes efficiently on block chain platforms.

Cost of Implementation: The significant initial financial investment required for EVMs with block chain.

Regulatory and Legal Frameworks: Establishing comprehensive legal frameworks for electronic voting, addressing liability, disputes, and technical failures.

Trust and Adoption: Overcoming public skepticism, political resistance, and concerns about foreign interference and manipulation.

Technical Challenges: Addressing software bugs, compatibility issues, and system failures in EVM and block chain integration**.**

REFERENCE:

Academic Journals: Search in academic databases like IEEE Explore, Google Scholar, or JSTOR using keywords such as "block chain," "electronic voting," and "EVM" to find scholarly articles and research papers.

News and Technology Websites: Websites like TechCrunch, Wired, and Coin telegraph often feature articles on the latest developments in electronic voting technology and block chain applications.

Government Reports: Government websites and agencies may release reports on pilot programs and research related to electronic voting and block chain. Look for reports from election commissions or technology departments.

Whitepapers: Some organizations and block chain projects publish whitepapers that detail their proposed solutions for electronic voting using block chain. These can provide in-depth information on the topic.

Books: Search for books on electronic voting systems and block chain technology at your local library or online retailers like Amazon.

Online Forums and Communities: Websites like Reddit and specialized forums often have discussions and links to relevant news and research articles.

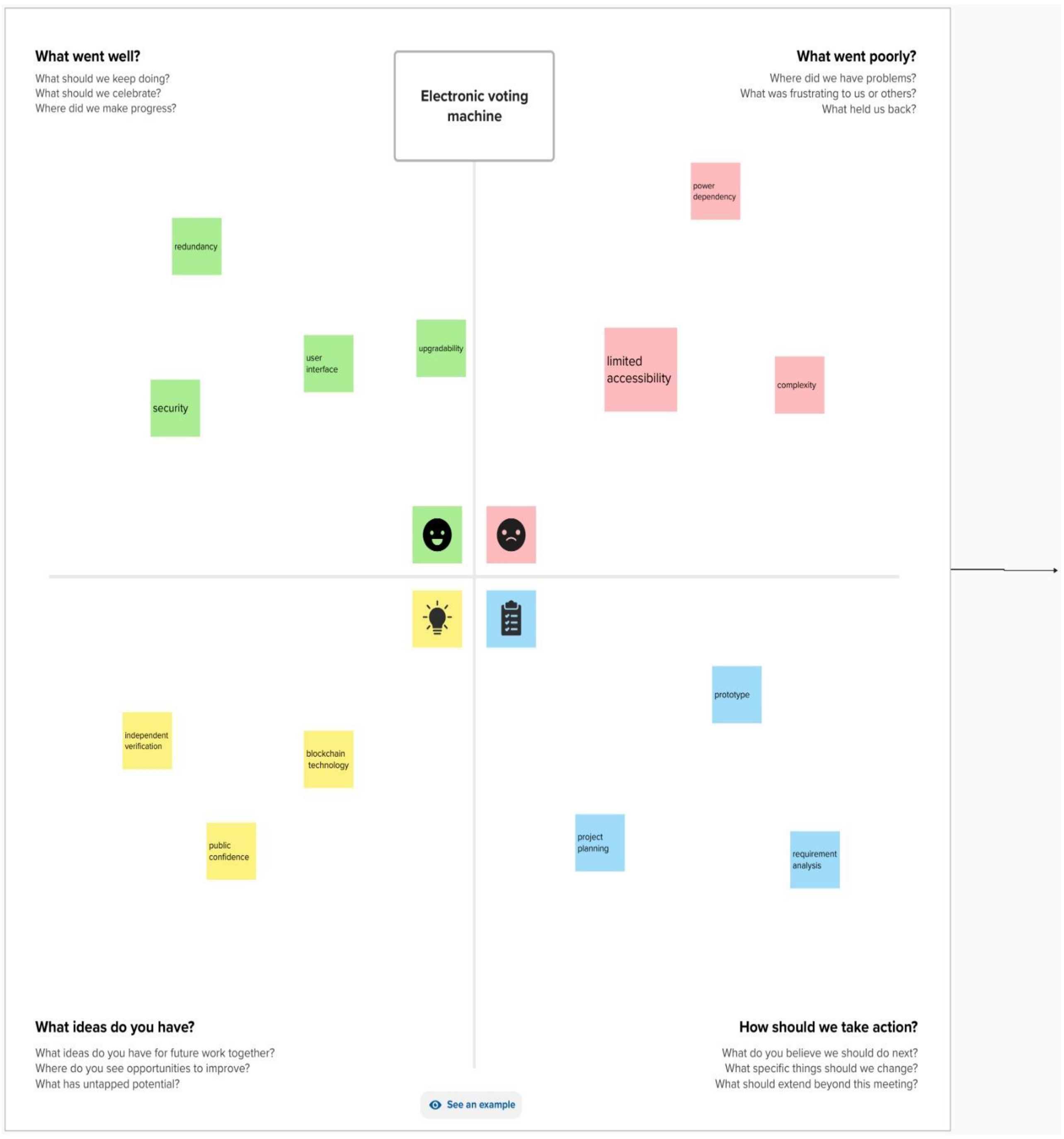
PROBLEM STATEMENT DEFINITION:

Real estate transactions and property records historically rely on paper-based documentation, fragmented data sources, and manual processes. This leads to significant inefficiencies, lack of transparency, and high costs associated with transferring property titles, recording deeds, coordinating between stakeholders, maintaining up-to-date records, and more.
With the rise of block chain technology, there is an opportunity to revolutionize real estate management by developing an open, distributed ledger for maintaining immutable property records and executing transactions through smart contracts. However, existing real estate workflows, regulations, and legacysystems pose challenges for adopting block chain solutions.
The problem is how to design and implement an enterprise-grade block chain network that interconnects relevant real estate stakeholders to streamline end-to-end real estate transactions. The network needs to reduce friction and Information asymmetry in real estate processes while complying with legal and regulatory requirements. Additionally, it needs to integrate with legacy databasesand IT systems used by incumbents like title companies, banks, and government agencies. This is a non-trivial task requiring expertise across real estate, block chain, security, law, and more.

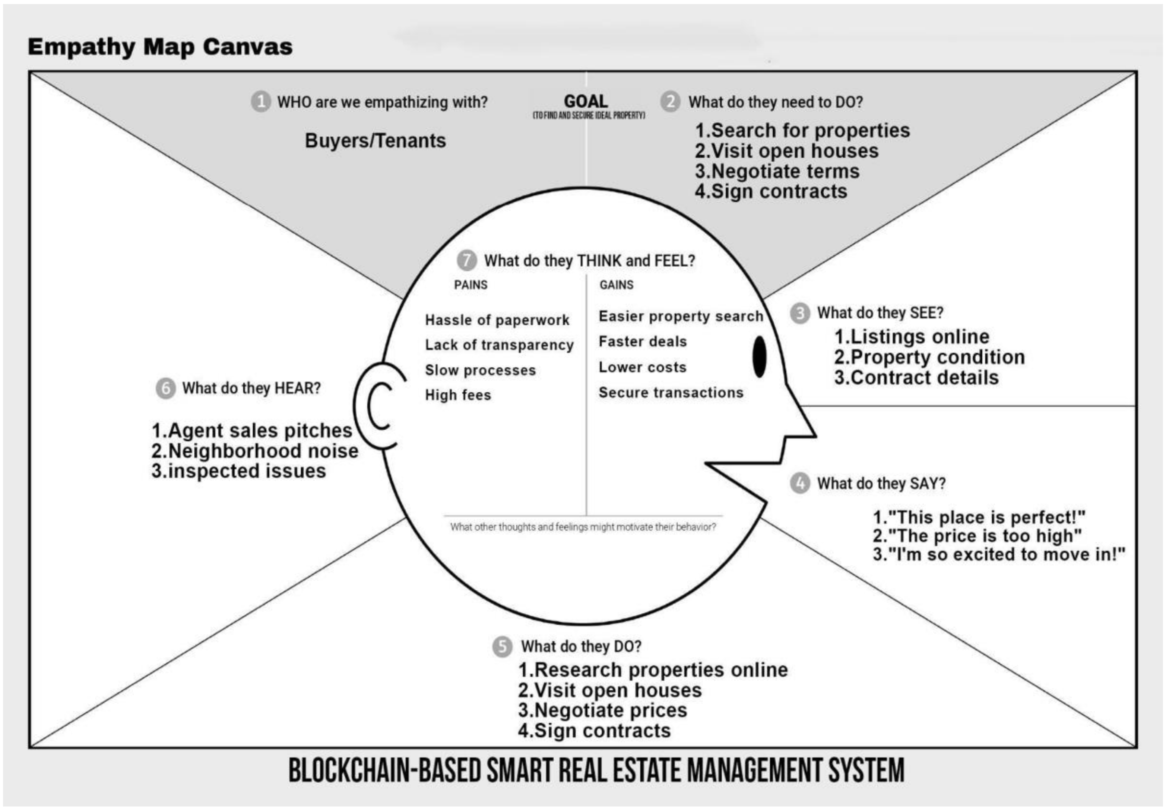
By building a block chain solution for real estate management, we can increase transparency, efficiency, and auditability while lowering costs and friction. This will benefit property buyers, sellers, renters, brokers, lenders, regulators, and all other stakeholders. The problem we aim to solve is developing the optimal combination of block chain technology, smart contracts, protocols, governance,and architecture to make this network a reality.

PROJECT IDEATION PHASE

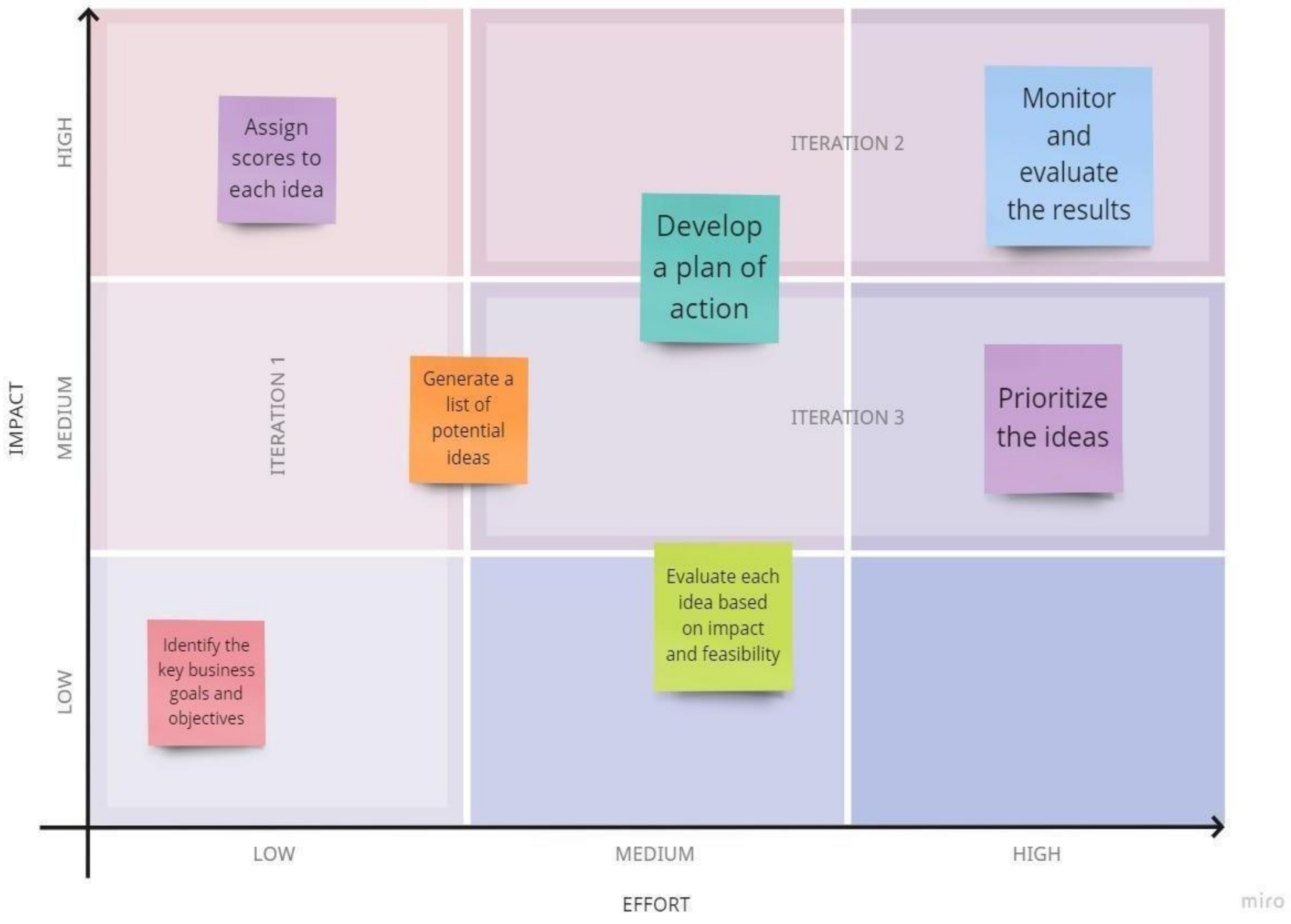
BRAINSTORMING:


What went well?
What should we keep doing?
What should we celebrate?
Where did we make progress?
Electronic voting machine
What went poorly?
Where did we have problems?
What was frustrating to us or others?
What held us back?
power dependency
redundancy
user interface
upgradability
limited accessibility
complexity
security
prototype
independent verification
blockchain technology
public confidence
project planning
requirement analysis
What ideas do you have?
What ideas do you have for future work together?
Where do you see opportunities to improve?
What has untapped potential?
How should we take action?
What do you believe we should do next?
What specific things should we change?
What should extend beyond this meeting?
See an example

EMPATHY MAP:



IDEA PRIORITIZATION:


IMPACT
HIGH
MEDIUM
LOW
EFFORT
LOW
MEDIUM
HIGH
ITERATION 1
ITERATION 2
ITERATION 3
Assign scores to each idea
Develop a plan of action
Monitor and evaluate the results
Generate a list of potential ideas
Prioritize the ideas
Identify the key business goals and objectives
Evaluate each idea based on impact and feasibility
miro

REQUIREMENT ANALYSIS:

Functional requirement

FR No.	**Functional Requirement (Epic)**	**Sub Requirement (Story / Sub-Task)**
FR-1	Real-Time Monitoring	Verify the identity of voters securely through biometrics or a unique identifier like a voter ID card.
FR-2	Voter casting	Allow voters to cast their votes electronically while maintaining their privacy. Ensure a user-friendly interface for voters, including those with disabilities.
FR-3	Block chain integration	Utilize block chain technology to store and record votes in a tamper-proof and transparent manner. Use a decentralized block chain network to prevent a single point of failure.
FR-4	User feedback and support	Establish channels for users to report issues, provide feedback, and seek technical support.

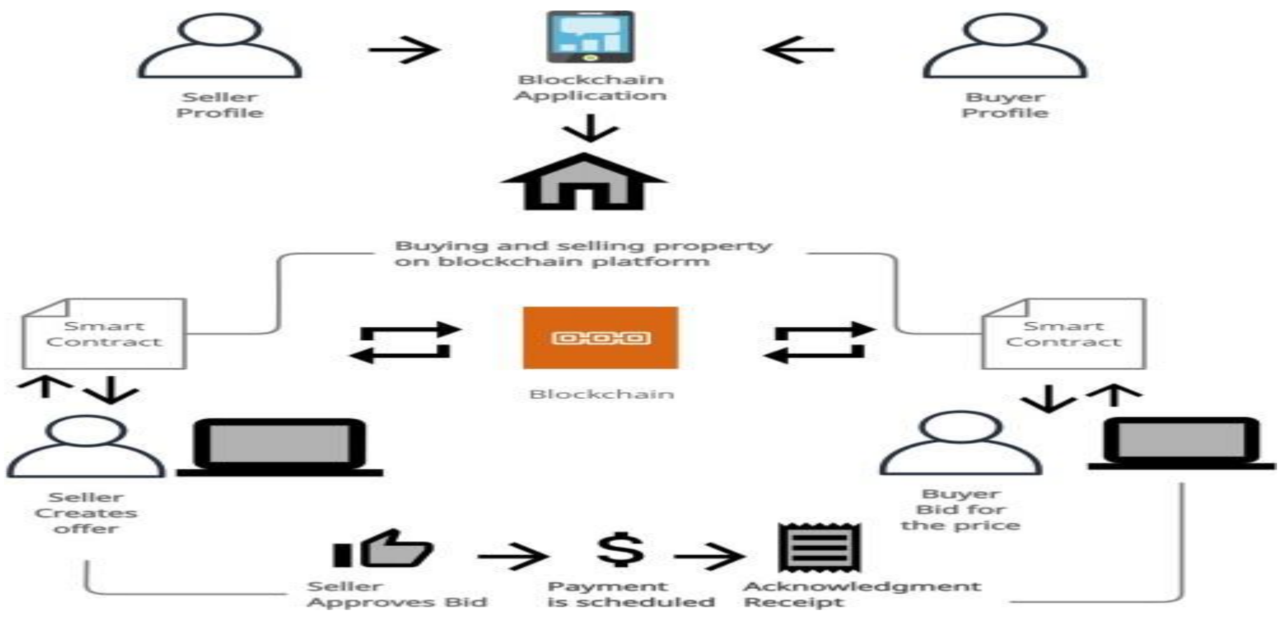
Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	Design a user-friendly interface for both voters and election officials, with clear and intuitive navigation. Ensure that the system is accessible to individuals with disabilities.
NFR-2	**Security**	The system should automatically be able authenticate all users with their unique username and password.
NFR-3	**Performance**	Ensure low latency in vote processing and result computation to maintain a smooth and efficient voting process. The system should be capable of handling peak loads during election hours without degradation in performance..
NFR-4	**Availability**	Information is restricted to each user's limited access.
NFR-5	**Scalability**	The system should be able to handle an increasing number of voters and transactions without a significant drop in performance.

PROJECT DESIGN:

DATA FLOW DIAGRAM & USER STORIES:



USER STORIES:

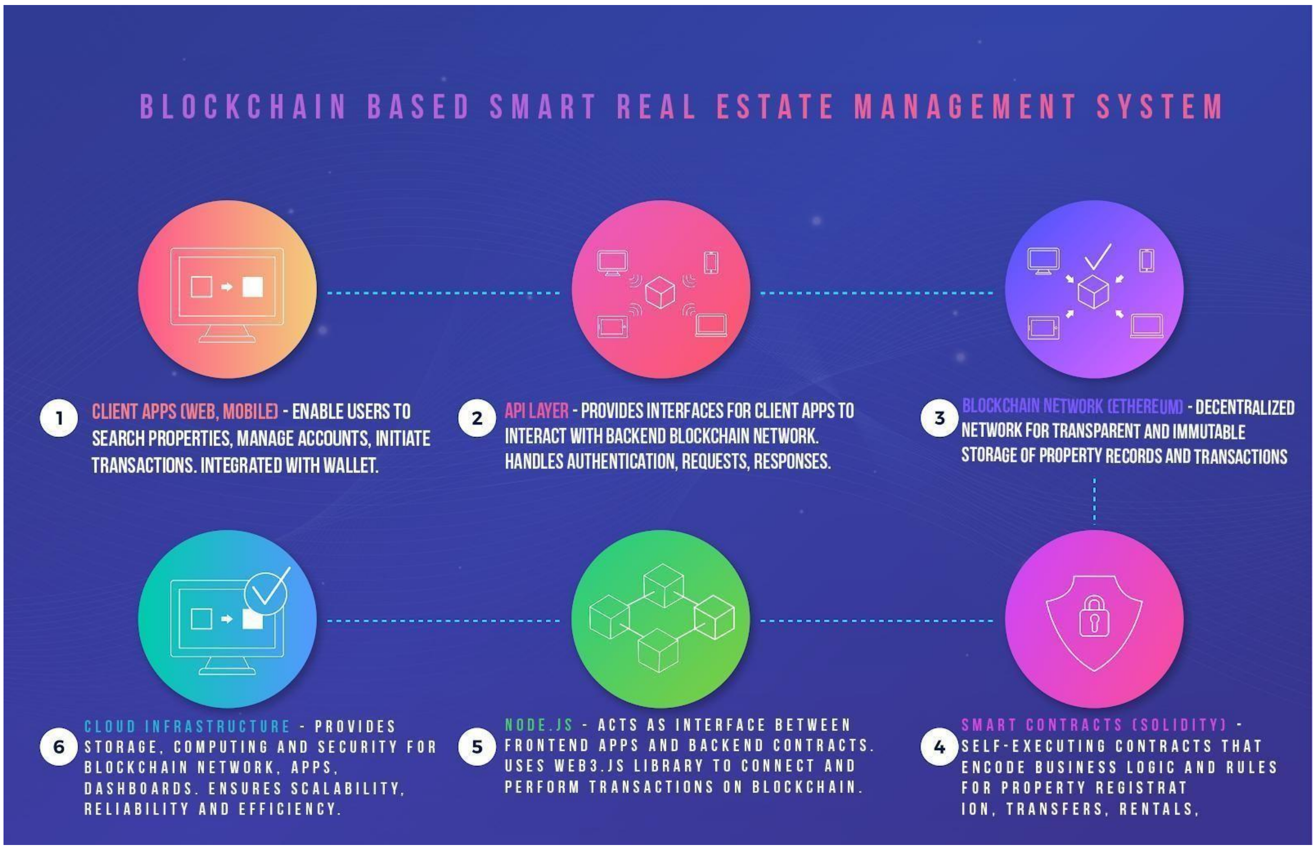
User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
Customer (Mobile user)	Registration	USN-1	As a voter, I want to be able to easily register for voting using an online platform linked to the block chain system, so I can participate in elections without the need to physically visit a registration office..	I can access my account / dashboard	High	Siva sree

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
	Verification Identity	USN-2	As an election authority, I want to verify the identity of voters securely and accurately to prevent fraudulent voting and ensure the integrity of the election.	I can receive confirmation email & click confirm	High	Subitha
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Santhanamari
		USN-4	As a user, I can register for the application through Gmail		Medium	Subitha
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sivasree
	Dashboard					
Customer (Web user)						
Customer Care Executive						
Administrator						

SOLUTION ARCHITECTURE:


BLOCKCHAIN BASED SMART REAL ESTATE MANAGEMENT SYSTEM
1 CLIENT APPS (WEB, MOBILE) - ENABLE USERS TO SEARCH PROPERTIES, MANAGE ACCOUNTS, INITIATE TRANSACTIONS. INTEGRATED WITH WALLET.
2 API LAYER - PROVIDES INTERFACES FOR CLIENT APPS TO INTERACT WITH BACKEND BLOCKCHAIN NETWORK. HANDLES AUTHENTICATION, REQUESTS, RESPONSES.
3 BLOCKCHAIN NETWORK (ETHEREUM) - DECENTRALIZED NETWORK FOR TRANSPARENT AND IMMUTABLE STORAGE OF PROPERTY RECORDS AND TRANSACTIONS
4 SMART CONTRACTS (SOLIDITY) - SELF-EXECUTING CONTRACTS THAT ENCODE BUSINESS LOGIC AND RULES FOR PROPERTY REGISTRAT ION, TRANSFERS, RENTALS,
5 NODE.JS - ACTS AS INTERFACE BETWEEN FRONTEND APPS AND BACKEND CONTRACTS. USES WEB3.JS LIBRARY TO CONNECT AND PERFORM TRANSACTIONS ON BLOCKCHAIN.
6 CLOUD INFRASTRUCTURE - PROVIDES STORAGE, COMPUTING AND SECURITY FOR BLOCKCHAIN NETWORK, APPS, DASHBOARDS. ENSURES SCALABILITY, RELIABILITY AND EFFICIENCY.

PROJECT PLANNING AND SCHEDULING:

TECHNICAL ARCHITECTURE:

<u>Hardware Components:</u>

Voting Terminal: The primary interface for voters to cast their votes. It should have a secure and tamper-resistant design to prevent physical tampering.

Secure Processing Unit (SPU): An isolated and secure hardware component that handles cryptographic operations, ensuring the integrity and security of the voting process.

Network Connectivity: The voting machines should be able to connect to a secure network for transmitting and receiving data.

<u>Software Components:</u>

Block chain: The core of the system, where all votes and transactions are recorded. A public or consortium block chain could be used to ensure transparency and security. Ethereal, Hyper ledger Fabric, or a custom block chain may be considered.

Smart Contracts: Smart contracts can be used to manage the voting process, ensuring that only eligible voters can cast their votes and recording the votes securely.

Voter Registration System: This system verifies and registers eligible voters. It may use biometrics or other secure methods for voter identification.

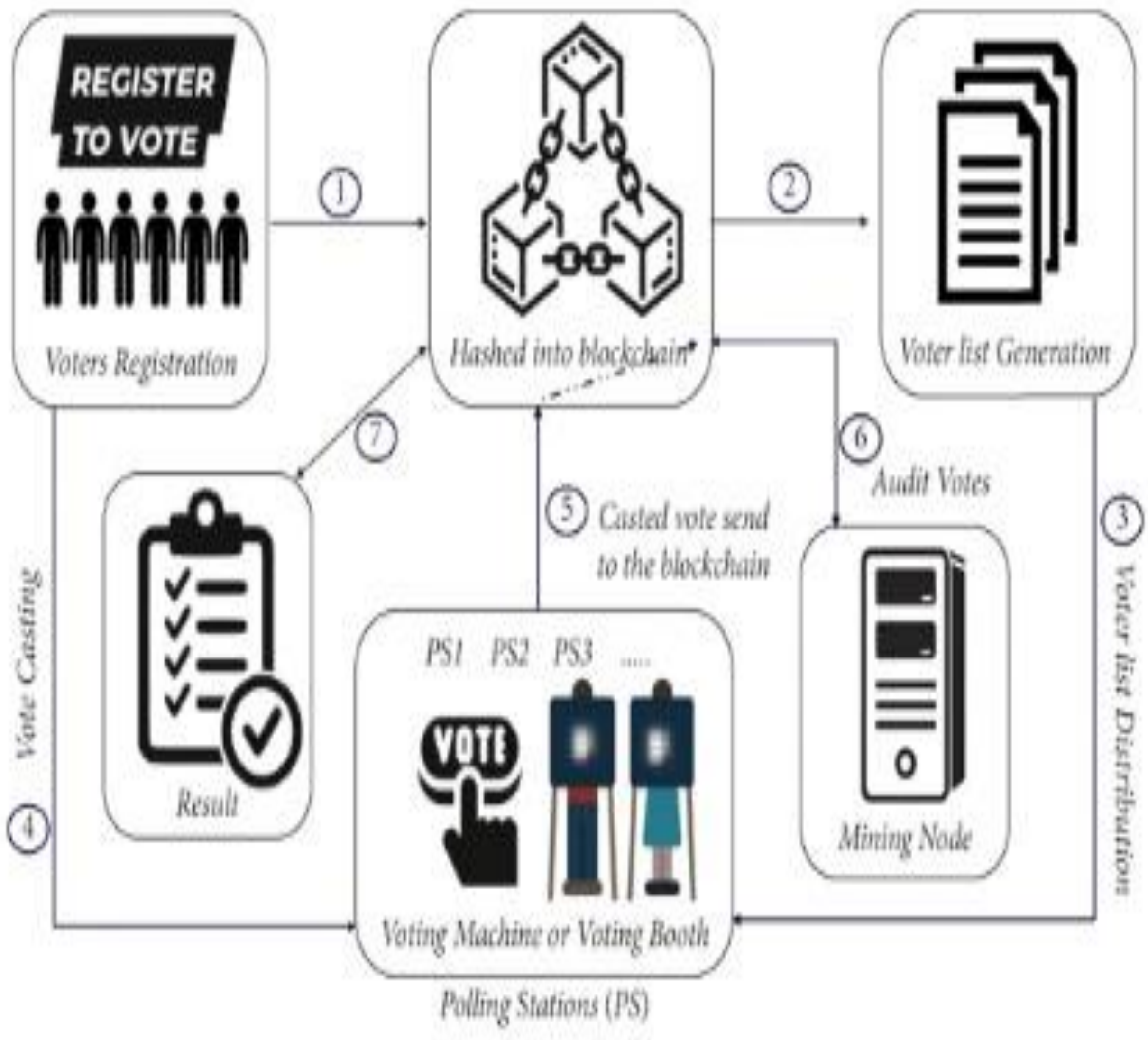
Work flow:

Voter Registration: Before the election, eligible voters must be registered securely. Their information is stored in the block chain, ensuring one person, one vote.

Casting Votes: On Election Day, voters use the EVM to cast their votes. The vote is encrypted and securely transmitted to the block chain.

Vote Verification: Voters can verify their votes on the block chain, ensuring their vote was counted accurately.

Tallying Votes: The block chain stores all votes, which can be counted and audited by authorized entities. The decentralized nature of the block chain ensures the security and transparency of the process.



CODING & SOLUTIONING

7.1 Feature 1

Smart contract (Solidity)

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

contract PropertyDetail

address public owner;


   struct Property {
      string propertyId;
      string name;
      string location;
      string discription;
      address currentOwner;
   }


   mapping(string => Property) public properties;
   mapping(address => mapping(string => bool)) public hasAccess;


   event PropertyAdded(
      string indexed propertyId,
      string name,
      string location,
      address indexed owner
);
event PropertyTransferred(
    string indexed propertyId,
    address indexed from,
    address indexed to
);


constructor() {
    owner = msg.sender;
}


modifier onlyOwner() {
    require(msg.sender == owner, "Only contract owner can call this");
    _;
}


modifier hasPropertyAccess(string memory propertyId) {
    require(
        hasAccess[msg.sender][propertyId],
        "You don't have access to this property"
    );
    _;
}
```

```
function  addProperty(
   string memory propertyId,
   string memory name,
   string memory location,
   string memory __description
) external onlyOwner {
   require(
      bytes(properties[propertyId].propertyId).length == 0,
      "Property already exists"
   );

   properties[propertyId]  =  Property({
      propertyId: propertyId,
      name: name,
      location: location,
      discription : __description,
      currentOwner: owner
   });

   hasAccess[owner][propertyId]  =  true;

   emit PropertyAdded(propertyId, name, location, owner);
}
```

```
    function transferProperty(
        string memory propertyId,
        address newOwner
    ) external hasPropertyAccess(propertyId) {
        require(newOwner != address(0), "Invalid new owner");

        address currentOwner = properties[propertyId].currentOwner;
        properties[propertyId].currentOwner = newOwner;

        hasAccess[currentOwner][propertyId] = false;
        hasAccess[newOwner][propertyId] = true;

        emit PropertyTransferred(propertyId, currentOwner, newOwner);
    }
    function getPropertyDetails(
        string memory propertyId
    ) external view returns (string memory, string memory, address) {
        Property memory prop = properties[propertyId];
        return (prop.name, prop.location, prop.currentOwner);
    }
}
```

This Solidity contract implements basic functionality to track real estate properties on the block chain. Here is an explanation of how it can be integrated into a real estate management system:

- The Property Detail contract allows the owner to add new properties to the Block chain by calling add property
Each property has a unique ID, name, location and description. The owneraddress
- A Property strict holds all the details of each property. These are stored in amapping that maps property IDs to Property struts'.
- The contract tracks who has access to each property through the has Access mapping. Only addresses with access can transfer the property.
- The transfer Property () function allows transferring a property to a new owneraddress. It requires the sender to have access, and updates the owner mapping.
- Events are emitted on property creation and transfer, so the block chain has animmutable record.
- Getter functions like getPropertyDetails () allow querying property data.

To integrate this into a full real estate system:

- The owner could be a real estate company that can add its listings via add Property ()
- Buyers would get access to a listed property, allowing them to transfer it tothemselves on purchase.
- Additional functions could encode more complex real estate workflows likefinancing, titling etc.
- The events created would provide traceability over the entire property lifecycle.
- The on-chain property data can be used by other smart contracts for automatingprocesses.

So in summary, this contract provides a basic reusable component to track core real estate data on-chain, which can be integrated into a larger blockchain ecosystem for the industry.

Contract ABI (Application Binary Interface):

The abs variable holds the ABI of an ethereal smart contract. ABIs are essential

Encoding and decoding function call the ethereum block chain

Meta Mask Check:

The code first checks whether the Meta Mask wallet extension is installed in the user's browser. If Meta Mask is not detected, it displays an alert notifying the user that Meta Mask is not found and provides a link to download it.

Ethers.js Configuration:

It imports the ethers library, which is a popular library for Ethereum development. It creates a provider using Web3 Provider, which connects to the user's Meta Mask wallet and provides access to Ethereum. It creates a signer to interact with the Ethereum block chain on behalf of the user. It defines an Ethereum contract address and sets up the contract object using ethers. Contract, allowing the JavaScript code to interact with the contract's functions. In summary, this code is used for interacting with an Ethereumsmart contract through Meta Mask and ethers.js. It configures the necessary Ethereum provider and signer for communication with the block chain and sets up a contract object for executing functions and fetching data from the specified contract address using the provided ABI

PERFORMANCE TESTING:

Voter Turnout: Measure the percentage of eligible voters who participate in the election using the EVM. Low voter turnout may indicate accessibility or usability issues.

Transaction Throughput: Evaluate the number of transactions (votes) the system can handle per unit of time, ensuring it can process votes efficiently during peak voting hours.

Latency: Measure the time it takes for a vote to be recorded on the blockchain. Lower latency is desirable to provide quick confirmation to voters.

Block chain Consensus Time: Assess the time it takes for a block to be mined and added to the block chain. Faster block confirmation enhances the real-time nature of the system.

Scalability: Evaluate the system's ability to handle an increasing number of voters and candidates without a significant decrease in performance.

Reliability and Availability: Measure the system's uptime and reliability during the election period. Downtime or unavailability can lead to disenfranchisement.

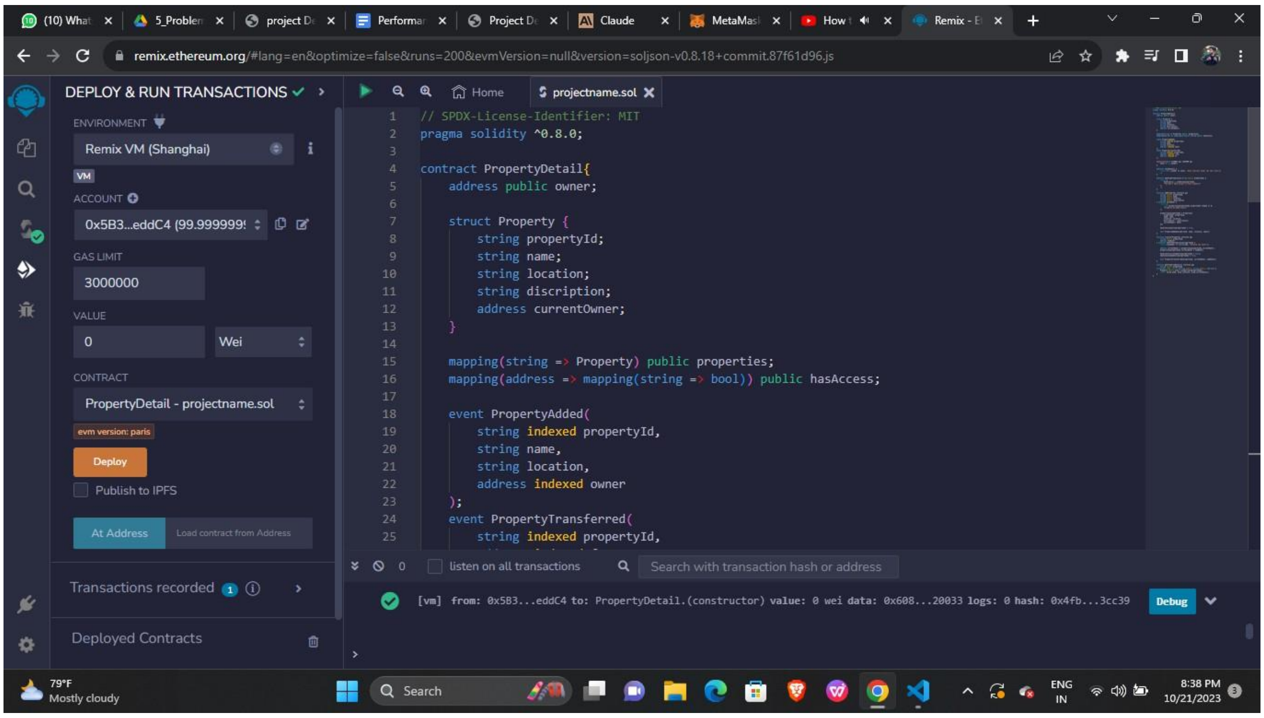
Security and Integrity: Monitor the system for any security breaches or attempted attacks. The absence of successful attacks is a critical performance metric.

Accuracy: Verify the accuracy of the election results by comparing them with other independent records or audits. Ensure that the EVM accurately reflects the will of the voters.

Block chain Size: Assess the growth of the block chain size over time and evaluate its impact on system performance and storage requirements.

RESULTS:

Output screen shots:

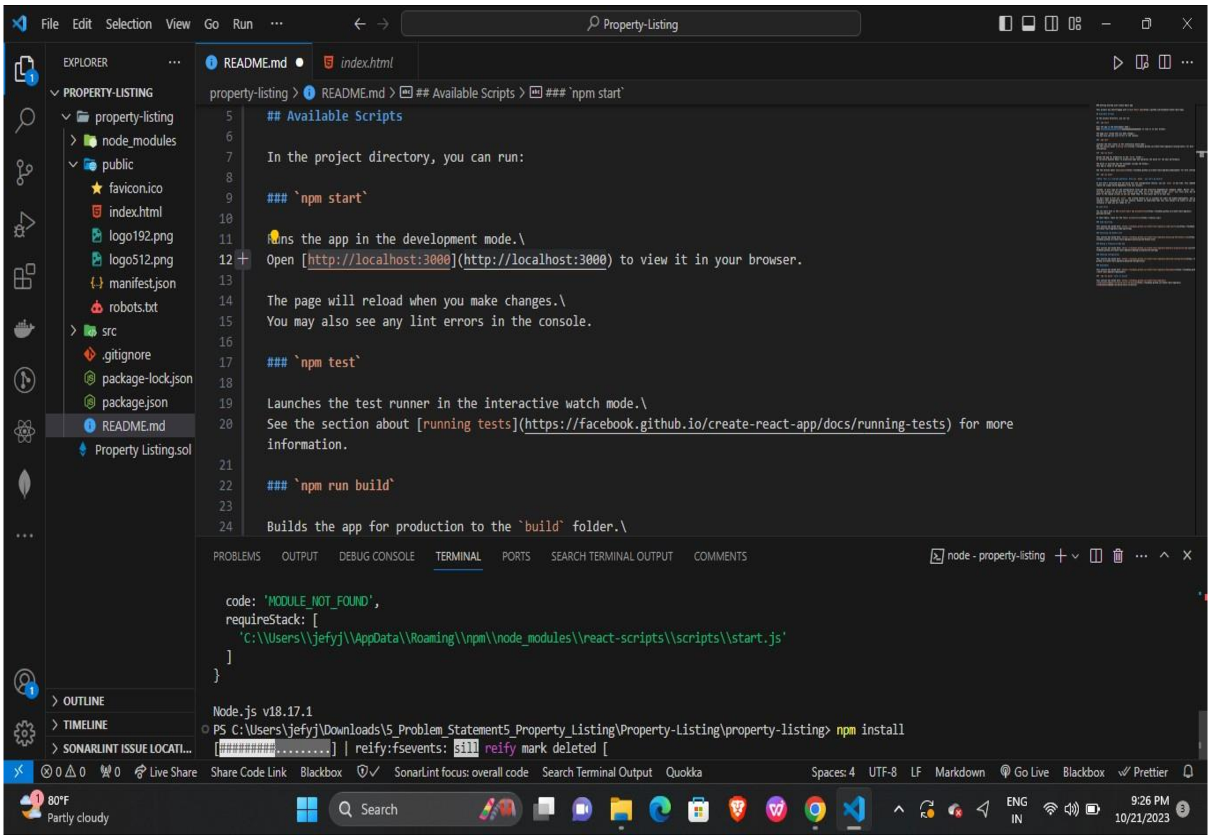


remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.18+commit.87f61d96.js

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

contract PropertyDetail{
    address public owner;

    struct Property {
        string propertyId;
        string name;
        string location;
        string discription;
        address currentOwner;
    }

    mapping(string => Property) public properties;
    mapping(address => mapping(string => bool)) public hasAccess;

    event PropertyAdded(
        string indexed propertyId,
        string name,
        string location,
        address indexed owner
    );
    event PropertyTransferred(
        string indexed propertyId,
```

```
[vm] from: 0x5B3...eddC4 to: PropertyDetail.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x4fb...3cc39
```

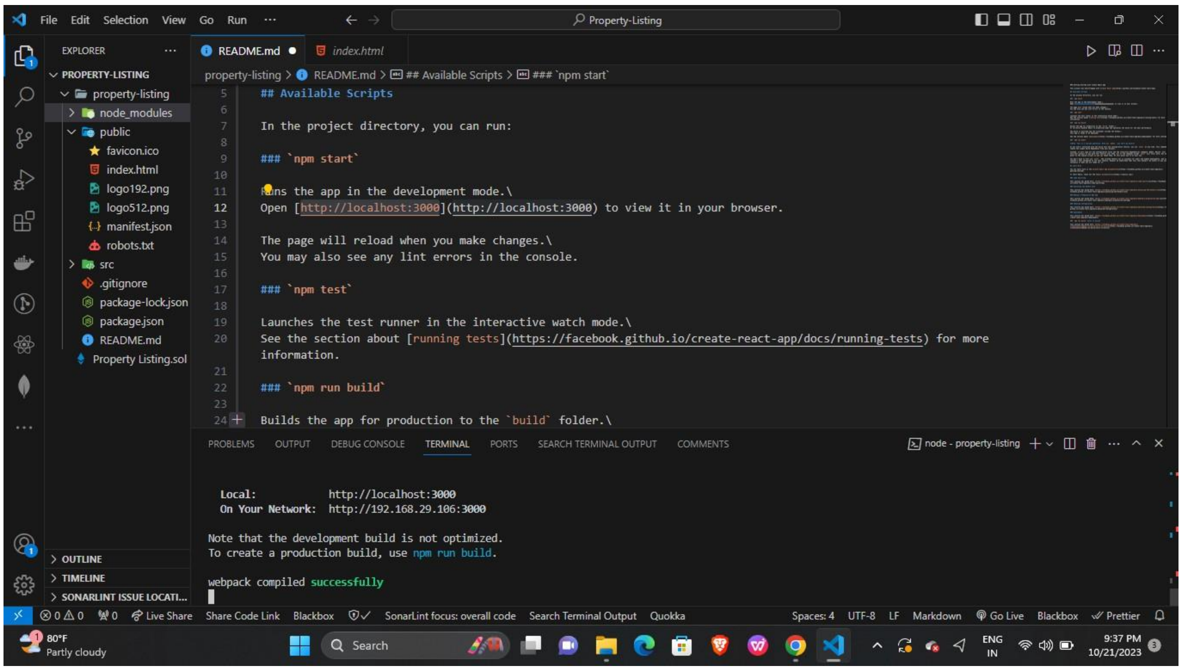
Creating a small contract:



```
## Available Scripts

In the project directory, you can run:

### `npm start`

Runs the app in the development mode.\
Open [http://localhost:3000](http://localhost:3000) to view it in your browser.

The page will reload when you make changes.\
You may also see any lint errors in the console.

### `npm test`

Launches the test runner in the interactive watch mode.\
See the section about [running tests](https://facebook.github.io/create-react-app/docs/running-tests) for more
information.

### `npm run build`

Builds the app for production to the `build` folder.\
```

```
  code: 'MODULE_NOT_FOUND',
  requireStack: [
    'C:\\Users\\jefyj\\AppData\\Roaming\\npm\\node_modules\\react-scripts\\scripts\\start.js'
  ]
}

Node.js v18.17.1
PS C:\Users\jefyj\Downloads\5_Problem_Statement5_Property_Listing\Property-Listing\property-listing> npm install
[#########.........] | reify:fsevents: sill reify mark deleted [
```

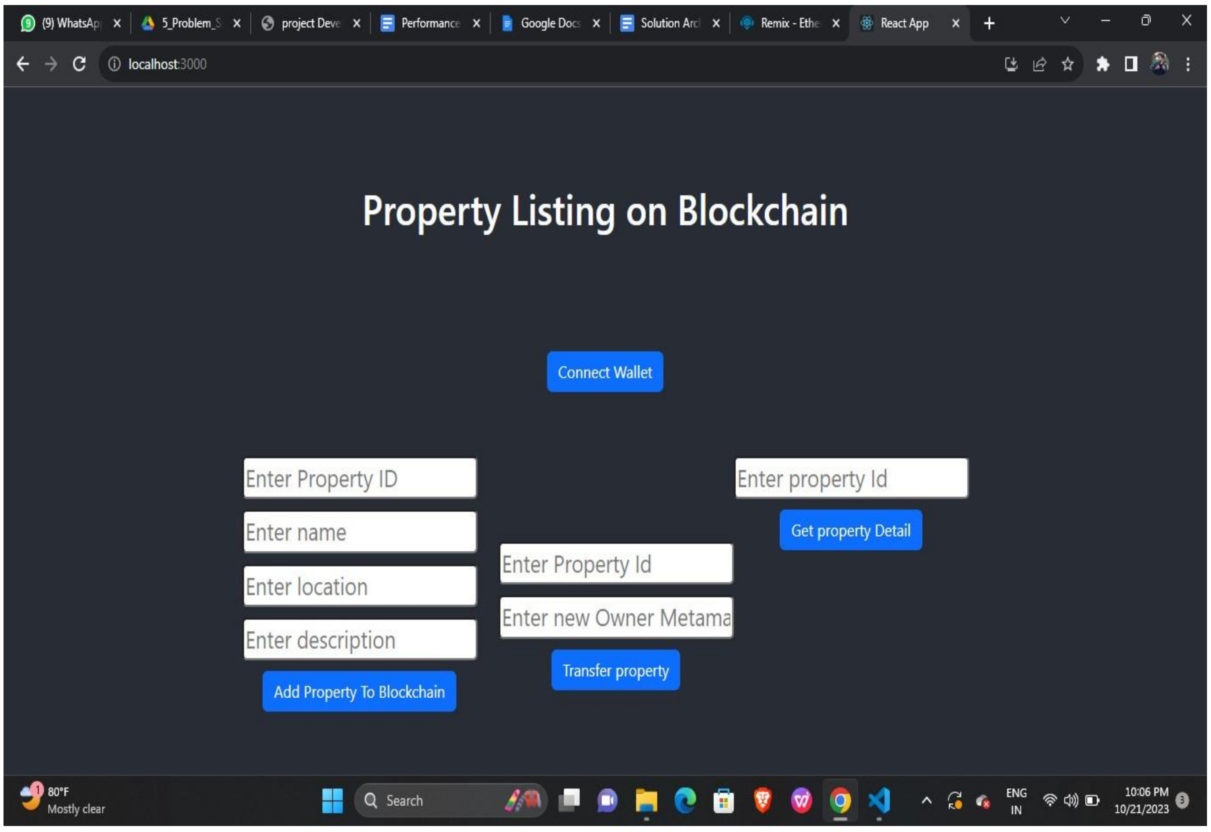
Installing dependencies:



```
## Available Scripts

In the project directory, you can run:

### `npm start`

Runs the app in the development mode.\
Open [http://localhost:3000](http://localhost:3000) to view it in your browser.

The page will reload when you make changes.\
You may also see any lint errors in the console.

### `npm test`

Launches the test runner in the interactive watch mode.\
See the section about [running tests](https://facebook.github.io/create-react-app/docs/running-tests) for more information.

### `npm run build`

Builds the app for production to the `build` folder.\
```

```
  Local:            http://localhost:3000
  On Your Network:  http://192.168.29.106:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Hosting the site locally:



Output screen

10. ADVANTAGES AND DISADVANTAGES

Advantages:

1. Security and Transparency:

Immutable Records: Block chain technology ensures that once a vote is recorded, it cannot be altered, providing a high level of security against tampering or fraud.

Transparency: The block chain ledger is publicly accessible, allowing anyone to verify the integrity of the voting process, thereby increasing trust in the system.

2. Elimination of Double Voting:

Block chain can prevent double voting by linking each voter's identity to a unique cryptographic key, which can only be used once.

3. Remote Voting:

EVMs with block chain can enable remote voting, making it easier for people who are unable to physically go to a polling station to cast their votes

4. Reduced Costs:

Electronic systems can potentially reduce the cost of printing and distributing paper ballots, and the manual effort required for counting.

5. Faster Results:

Counting and reporting results can be done in real-time or much faster, reducing the time needed to determine the outcome of an election.

Disadvantages:

1. Security Risks:

Although block chain is generally secure, it is not entirely immune to attacks or vulnerabilities. If the block chain or the EVM software has security flaws, the system could still be compromised.

2. Digital Divide:

Electronic voting systems may exclude individuals who do not have access to the necessary technology or are not comfortable using it, potentially exacerbating the digital divide.

Privacy Concerns:

Block chain voting systems may raise privacy concerns if they require a complete public ledger of votes. Ensuring anonymity while still providing transparency can be a challenging balance to strike.

3. Voter Authentication:
   Verifying the identity of voters in a remote voting system is a significant challenge. Ensuring that voters are who they claim to be is crucial to prevent fraud.
4. Complexity:
   Implementing and maintaining a block chain-based voting system can be complex and costly. It may require a high level of technical expertise, and many governments may struggle with the transition.
5. Voter Trust:
   The adoption of new technology can be met with scepticism and mistrust, especially when it comes to something as critical as elections. Building trust in the system may take time.
6. Legal and Regulatory Challenges:
   Adapting legal frameworks and regulations to accommodate block chain-based voting can be challenging. There may be disagreements and uncertainties about how to handle disputes or audits.
7. Voting Irregularities:
   While block chain can prevent certain types of fraud, it cannot entirely eliminate all voting irregularities, such as voter intimidation or coercion.

11. CONCLUSION:

In conclusion, electronic voting machines using block chain technology offer a promising solution to many of the challenges associated with traditional voting systems. The advantages of enhanced security, transparency, and efficiency are evident, with the potential to eliminate double voting and reduce costs. However, these systems also come with their own set of disadvantages and concerns.

The key to successfully implementing electronic voting machines with block chain lies in addressing these disadvantages and mitigating potential risks. Security vulnerabilities, the digital divide, privacy concerns, and the need for robust voter authentication systems all require careful consideration. Additionally, building trust in this new technology and navigating legal and regulatory challenges are essential components of a successful transit. While electronic voting machines using block chain have the potential to revolutionize the way we conduct elections, the path forward should be characterized by thorough testing, transparency in the development and implementation process, and a commitment to continuously improving and securing these systems. With the right strategies and safeguards in place, these technologies may pave the way for more secure, efficient, and accessible democratic processes in the future.

12. FUTURE SCOPE:

Enhanced Security: Block chain technology can provide a robust and tamper-proof foundation for voting systems. As block chain technology continues to advance and become more secure, the future scope of EVMs is likely to see even stronger protection against hacking, fraud, and manipulation.

Global Adoption: While the adoption of block chain-based EVMs is still in its early stages, the future scope includes the potential for widespread adoption around the world. As more countries experiment with and implement this technology successfully, others are likely to follow suit.

Remote Voting: With the continued development of secure identity verification methods, EVMs can enable remote voting on a larger scale. This has the potential to increase voter turnout and accessibility for individuals who face barriers to physical polling places.

Accessibility: Future EVMs can be designed with user-friendly interfaces to cater to a diverse range of voters, including those with disabilities and language barriers, ensuring inclusivity.

Transparency: The transparency and auditability of block chain technology can lead to greater trust in election outcomes. The future scope may include a more engaged and informed electorate, as voters can verify the integrity of the system.

Reduced Costs: As technology evolves and becomes more streamlined, the costs associated with implementing and maintaining EVMs using block chain may decrease, making them more financially viable for governments.

Interoperability: EVMs could become more interoperable, allowing for data sharing and standardization across different voting systems. This could promote consistency and compatibility between regions and countries.

Immutable Records: The immutability of blockchain records can provide a valuable historical record of election results. This could be used for historical analysis and auditing.

Legal Frameworks: As block chain-based voting systems become more prevalent, legal and regulatory frameworks are likely to evolve to address issues related to disputes, auditing, and the handling of irregularities.

APPENDIX:

SOURCE CODE:

```
import java.io.*;
import java.util.*;

interface client
{
    void vote();
    void result();
}

class server1 implements client
{
    private int count1=0;
    private int count2=0;
    private int count3=0;
    private int count4=0;
    private int total=0;

    public void vote()
    {
        int b=1;
        while(b!=0)
        {
            System.out.println("welcome to the voting of IIITN General
Secratory election");
            System.out.println("\n" + "Press 1 for Voting OR 0 to stop
voting");
            Scanner s3 =new Scanner(System.in);
            b=s3.nextInt();

            switch(b)
            {
                case 1 :
                System.out.println();
                System.out.println("ENTER YOUR COLLEGE ID in format of
BT21....");
                Scanner s= new Scanner(System.in);
                String ID = s.nextLine();
                int temp=1;
                try
                {


                    File tt = new File("temp.txt");
                    FileReader fr1=new FileReader(tt);
                    BufferedReader br1=new BufferedReader(fr1);

                    String st3;

                    while((st3=br1.readLine())!=null)
                    {
                        if(st3.contains(ID))
                        {
                            temp=0;
                        }
                    }
                }
                catch(IOException i)
                {
                    System.out.println("ERROR");
                }


                if(temp!=1)
                {
                    System.out.println("Already voted ! ");
                    continue;
                }
                else
                {
                    try
                    {
                        FileWriter fw2 = new
FileWriter("temp.txt",true);
                        fw2.write(ID);
                        fw2.close();
                    }
                    catch(IOException i)
                    {
                        System.out.println("ERROR");
                    }
                    try
                    {
                    File input =new File("list.txt");
                    FileReader fr=new FileReader(input);
                    BufferedReader br=new BufferedReader(fr);

                    String str;
                    int flag=0;
                    while((str=br.readLine())!=null)
                    {
                        if(str.contains(ID))
                        {
                            System.out.println(str);
                            flag=1;
                        }
                    }
                    if(flag!=1)
                    {
                        System.out.println("SORRY....Your ID is not in
a voter list, You can't vote");
                        continue;
                    }
                    fr.close();
                    }
                    catch(IOException e)
                    {
                        System.out.println("ERROR :-there would be some
problem in reading of a file");
                    }

                    System.out.println("... -: Candidates are :- ..." +
"\n");
                    System.out.println("1- ADITYA SINGH");
                    System.out.println("2- YASH MISHRA");
                    System.out.println("3- ANSHUMAN DAS");
                    System.out.println("4- NOTA"+"\n");

                    System.out.println("please select any one of
them");
                    int a;
                    Scanner s2=new Scanner(System.in);
                    a=s2.nextInt();



                    if(a==1)
                    {
                        count1++;
                        total++;
                    }
                    else if(a==2)
                    {
                        count2++;
                        total++;
                    }
                    else if(a==3)
                    {
                        count3++;
                        total++;
                    }
                    else
                    {
                        count4++;
                        total++;
                    }
                    break;
                }

                case 0 :
                try
                {
                    FileWriter fw = new FileWriter("temp.txt", false);
                    PrintWriter pw = new PrintWriter(fw, false);
                    pw.flush();
                    pw.close();
                }
                catch(IOException i)
                {
                    System.out.println(i);
                }
                try
                {
                    FileWriter fw =new FileWriter("votecount.txt");
                    fw.write("votes for ADITYA SINGH are :- " + count1
+ "\n"+"votes for YASH MISHRA are :- "+ count2 + "\n"+
                            "votes for ANSHUMAN DAS are :- "+ count3
+"\n"+"votes for NOTA are :- "+ count4 +"\n");
                    fw.write("voting percentages are :- " +
total*20+"%");

                    fw.close();
                }
                catch(IOException i)
                {
                    System.out.println("ERROR:- ISSUE IN SECOND FILE 
WRITING");
                }
                break;

                default :
                System.out.println("Invalid Input");
                break;
            }
        }

    }


    public void result()
    {
        System.out.println("results are");
        try
        {

            File res= new File("votecount.txt");
            FileReader r=new FileReader(res);
            BufferedReader b=new BufferedReader(r);

            String str2;
            while((str2=b.readLine())!=null)
            {
                System.out.println(str2);
            }

            r.close();


        }
        catch(IOException i)
        {
            System.out.println("ERROR:- issue in a second file 
reading");
        }

    }
}

class VoterList
{
    void listcreation()
    {

        try
        {
            FileWriter f=new FileWriter("list.txt");


            String arr[]={"BT21CSE171 DEEPAK SINGH CHAUHAN","BT21CSE179 
AAYUSH PATIL","BT21CSE206 PRIYANSHU SINGH"
```

```
            ,"BT21CSE200 VAIBHAV TAYVADE","BT21CSE131 PRANAV CHANDAK"};

            int length=arr.length;
            for(int i=0;i<length;i++)
            {
                f.write(arr[i]+"\n");

            }
             f.close();

        }
        catch(IOException i)
        {
            System.out.println("ERROR:-there would be a problem in
voter list creation");
        }
    }
}


class EVM
{
    public static void main( String args[])
    {
        VoterList obj2= new VoterList();
        obj2.listcreation();
        server1 obj=new server1();
        obj.vote();

        System.out.println("thanks for voting");
    }
}
```

GitHub link:

https://github.com/Subhi1710V/NM2023TMID10674/tree/main

Demo video link: