Project Report

Team ID	NM2023TMID05236
Project Name	ELECTRONIC VOTING SYSTEM

Submitted by

TEAM LEADER : KAVIYAPRIYA S

TEAM MEMBER 1 :SANDHIYA P

TEAM MEMBER 2 : VARSHINI S

TEAM MEMBER 3 : GOPIKA A

PROJECT REPORT FORMAT

1.INTRODUCTION

An electronic voting system (EVS) is a method of voting that uses electronic means to either aid or take care of casting and counting votes. Electronic voting technology can include punched cards, optical scan voting systems, or specialized voting kiosks. More recently, it involves internet voting systems, where voters can cast their ballots online through secure websites. EVS aims to make the voting process more efficient, convenient, and accessible to a larger number of people. However, the implementation of electronic voting systems raises concerns about security, integrity, and privacy, which need to be carefully addressed to ensure the credibility of the electoral process

1.1 Project overview :

An electronic voting system based on blockchain technology offers a secure, transparent, and tamper-proof way to conduct elections. In this system, the entire voting process, from the registration of voters to the tallying of votes, is recorded in a blockchain. Here's an overview of how it works1. Voter Registration Voters are registered in the blockchain with a unique digital identity. This identity is verified through various methods to ensure the eligibility of the voter.2. Voting Process: When an election occurs, candidates and their respective information are stored in the blockchain. Voters can securely cast their votes through an online platform using cryptographic keys. Each vote is recorded as a transaction on the blockchain.

1.2 Purpose

The purpose of an electronic voting system is to streamline and enhance the voting process by using electronic means to cast and count votes. Electronic voting systems are designed to increase efficiency, accuracy, and accessibility in elections. They aim to Electronic voting systems can speed up the voting process, reducing long lines and wait times at polling stations. Electronic systems can minimize errors in vote counting, reducing the chances of miscounted or invalidated votes. Electronic voting can be more accessible for people with disabilities, providing options for audio or visual assistance.

2.LITERATURE SURVEY

1.Blockchain for electronic voting system

Implementing blockchain technology in an electronic voting system can enhance security, transparency, and integrity. Each vote could be recorded as a transaction on the blockchain, ensuring immutability. Additionally, the decentralized nature of blockchain reduces the risk of tampering, making it difficult for any single entity to manipulate the results. Smart contracts could automate the process, ensuring that the rules of the election are followed precisely. However, challenges such as scalability and ensuring voter privacy still need to be carefully addressed for a successful implementation.

2.A Blockchain-Based Approach for electronic voting system in Healthcare Supply Chain

Implementing a blockchain-based approach for electronic voting in the healthcare supply chain can enhance transparency, security, and traceability. By utilizing blockchain technology, you can create a decentralized and tamper-proof system where all voting transactions are recorded in a public ledger. This ensures the integrity of the voting process and helps prevent fraud or manipulation.

Additionally, smart contracts can be employed to automate the voting process, ensuring that predefined rules are followed, and results are accurately calculated. Participants in the healthcare supply chain, such as hospitals, pharmaceutical companies, and regulatory bodies, can securely cast their votes and view the transparent results in real-time.

3. A semantic blockchain-based system for electronic voting system

A semantic blockchain-based system for electronic voting combines blockchain technology with semantic web technologies to enhance the security, transparency, and efficiency of electronic voting processes. By using smart contracts and semantic data, it ensures the integrity of the voting system and enables automated verification of votes. Smart contracts can facilitate the creation of tamper-proof, self-executing agreements, ensuring that the voting process is carried out as intended. Semantic web technologies allow for a more meaningful understanding of the data, enabling better validation of voter eligibility and ensuring accurate counting of votes. Overall, this system provides a secure and trustworthy platform for electronic voting, addressing many of the challenges associated with traditional voting methods Khizar

2.1 Existing problem

Electronic voting systems based on blockchain technology aim to enhance security, transparency, and accessibility in elections. However, there are several existing challenges and concerns associated with these systems. Ensuring voter privacy while maintaining the transparency of the blockchain is a complex task. It's crucial to prevent any link between the voter and their vote while still verifying the authenticity of the vote. Blockchain-based electronic voting systems are not immune to cyberattacks. If an attacker gains control of a majority of the network's nodes, they can manipulate the voting results. Ensuring a robust and decentralized network is essential. Verifying the identity of voters in an online environment is challenging. Without proper authentication methods, there's a risk of unauthorized or fraudulent voting. Many voters may not be familiar with blockchain technology or how to use electronic voting systems. Ensuring that voters are educated and comfortable with the technology is crucial to prevent confusion or mistakes during the voting process. While immutability is a strength of blockchain, it becomes a problem if there are errors in the recorded votes. Once the data is on the blockchain, it's difficult to change, making it challenging to rectify mistakes or address any discrepancies.

2.2 References

1. "Voting: What Is, What Could Be" by Philip B. Stark and David L. Dill - This book provides a comprehensive overview of electronic voting systems, their challenges, and potential solutions.

2. "Security Engineering: A Guide to Building Dependable Distributed Systems" by Ross J. Anderson - This book covers various aspects of security engineering, including electronic voting systems, and is a valuable resource for understanding the security issues related to electronic voting.

3. "Electronic Voting: First International Joint Conference, E-Vote-ID 2016, Bregenz, Austria, October 18-21, 2016, Proceedings" edited by Melanie Volkamer, VoteID 2016 - This conference proceedings volume contains research papers and studies presented by experts in the field of electronic voting.

4. IEEE Xplore Digital Library- IEEE Xplore is a digital library that provides access to a vast collection of articles, conference papers, and standards related to electronic voting systems. You can search for specific topics within the field of electronic voting.

5. Google Scholar - Google Scholar is a freely accessible web search engine that indexes the full text or metadata of scholarly literature across an array of publishing formats and disciplines. You can use it to find academic papers, theses, and articles related to electronic voting systems.

2.3 Problem Statement Definition

1. LIMITED INFRASTRUCTURE AND PRODUCTION CAPABILITIES:

Major Pharmaceutical companies does not invest and establish production units in developing countriesdue to geopolitics, market inaccessibility and government instabilities. They are more focused onmanufacturing and circulation of branded medicines in developed countries like USA and Europe due to percapita income and pricing monopoly. Developing countries faces major challenges on pharmaceuticalindustries investment due to poor infrastructure and lack of government funds forresearch and infrastructureimprovements. Existing pharmaceutical units are struggling to meeting global standards due to need of heavyinvestments in new production and packaging machineries.

2.AMBIGUOUS REGULATIONS

Regulatory obligation plays a vital role to implement track and trace system for traceability. Developed countries like US in 2018 and Europe in 2019 implemented serialization compliance successfullyfor drug traceability. Before making digital traceability an obligatory regulation, DSCSA a drug controllingbody of FDA runs pilot programs with joint initiative of drug manufacturers, wholesale distributors andcommunity pharmacists.

3.INSUFFICIENT TECHNO-FUNCTIONAL RESOURCES FOR IMPLEMENTING ANDSUSTAINING ELECTRONIC VOTING SYSTEM

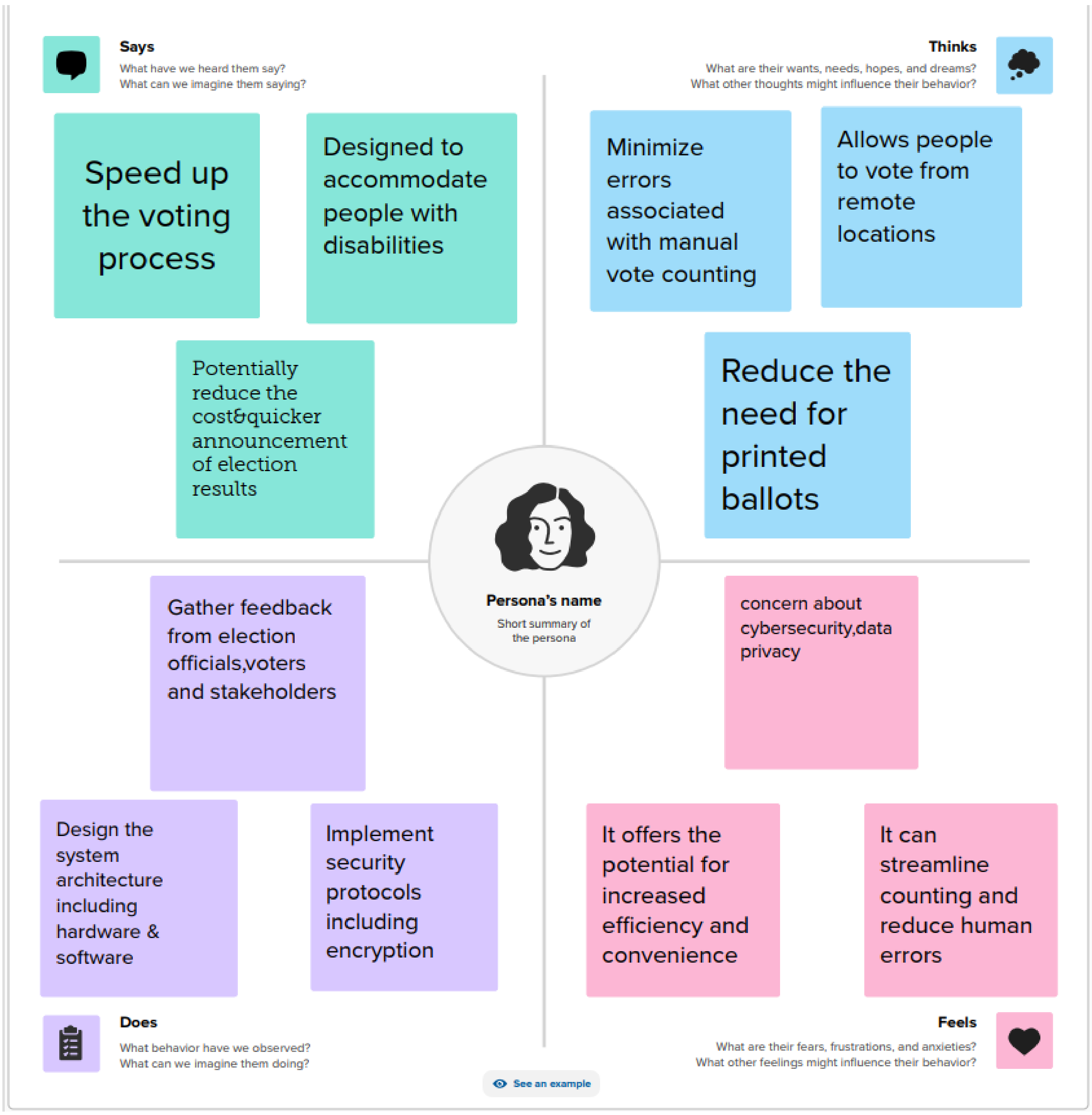
counterfeiting a serious threat to public health. It is a collective job for every stakeholder in supply chain to prevent counterfeiting and illicit. Implementing and sustaining serialization system for required skillful resources. Serialization and electronic voting system processes involve packaging lines, Barcode readers, scanner, label grading system, site and global level serialization system which can handle globally. Any human, mechanical and technical error can cause adversely to human life.

4.UNSECURE AND UNRELIABLE TECHNICAL INFRASTRUCTURE FOR ELECTRONIC VOTING SYSTEM

Technology advancement is another main challenge in developing countries for authenticating and tracing pharmaceutical products digitally. In recent years, some developed countries like US and Europe haveadopted serialization regulation under which manufacturer require to print a unique identifier printedwith the 2D barcode on individual drug unit. This unique identifier is key source for authenticating drug andtracing its origin of manufacturing. Printing unique identifier and keeping its key data in repository requiredspecial packaging equipment, tamper proof seals and global traceability software.

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



Says

What have we heard them say?
What can we imagine them saying?

Speed up the voting process

Designed to accommodate people with disabilities

Potentially reduce the cost&quicker announcement of election results

Thinks

What are their wants, needs, hopes, and dreams?
What other thoughts might influence their behavior?

Minimize errors associated with manual vote counting

Allows people to vote from remote locations

Reduce the need for printed ballots

Persona's name

Short summary of the persona

Does

What behavior have we observed?
What can we imagine them doing?

Gather feedback from election officials,voters and stakeholders

Design the system architecture including hardware & software

Implement security protocols including encryption

Feels

What are their fears, frustrations, and anxieties?
What other feelings might influence their behavior?

concern about cybersecurity,data privacy

It offers the potential for increased efficiency and convenience

It can streamline counting and reduce human errors

as

3.2 Ideation & Brainstorming

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!

S.Kaviyapriya

- Secure and transparent voting system using blockchain technology to ensure tamper proof records
- Absolute Anonymity and Transparency
- Regular security audits

P.Sandhiya

- creating user friendly mobile applications and accessibility for wide range of people
- Zero cost implementation
- Multi-Factor Aunthentication

S.Varshini

- Encrypted Voting Data
- Eliminating voter suppression
- using biometric data like fingerprints or facial recognition

A.Gopika

- Accessible voting station
- creating a system that is completely immune to hacking
- paper trail verification

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Regular security audits

using biometric data like fingerprints or facial recognition

paper trail verification

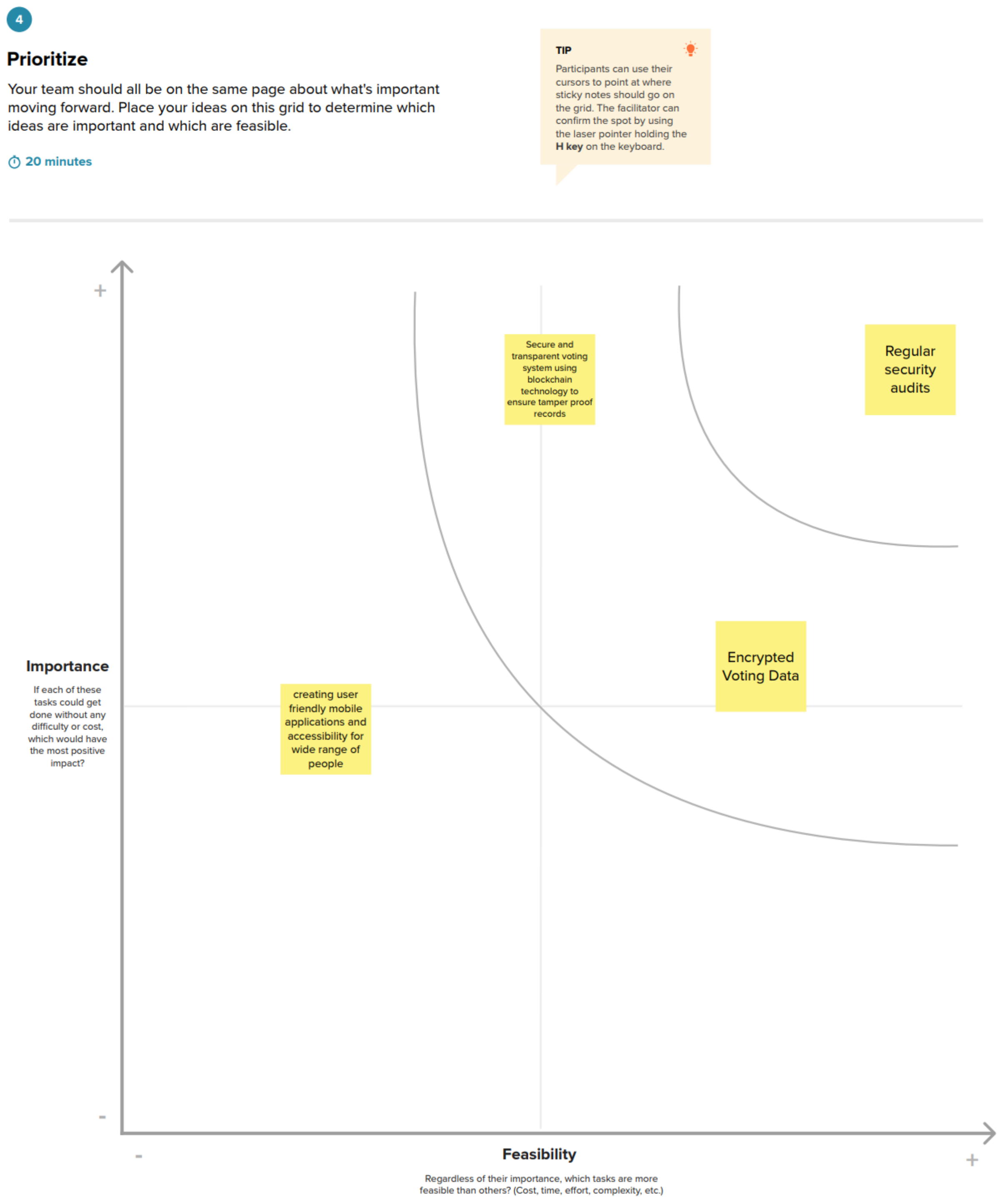
Accessible voting station

Secure and transparent voting system using blockchain technology to ensure tamper proof records

Encrypted Voting Data

Multi-Factor Aunthentication

creating user friendly mobile applications and accessibility for wide range of people


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

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.

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

+
-
-
+
Secure and transparent voting system using blockchain technology to ensure tamper proof records
Regular security audits
Encrypted Voting Data
creating user friendly mobile applications and accessibility for wide range of people

4.REQUIREMENT ANALYSIS

4.1 Functional requirement

FR No.	Functional Requirement	Description
FR- 1	Voting Creation and Registration	Users should be able to create and register voting on the Ethereum blockchain. voting registration should include metadata such as title, description, author, date, and any other relevant information.
FR- 2	VotingStorage and Encryption	Voting should be securely stored on the blockchain, with data encryption to protect their integrity and confidentiality.
FR- 3	Drug Tracking and Metadata Management	Users should have the ability to update drug metadata, including tags, categories, and descriptions. The system should support searching and filtering voting based on metadata.
FR- 4	voting Ownership and Transfer	voting should be associated with specific owners, and ownership should be transferable through blockchain transactions. Ownership transfers should be securely recorded on the blockchain.
FR- 5	Access Control and Permissions	Define access control and permissions for drug viewing, editing, and transfer. Implement role-based access control (RBAC) for different users or user groups.
FR- 6	Smart Contracts:	Utilize smart contracts for managing voting ownership, transfers, and permissions. Implement contract functionality for executing predefined rules and logic.
FR- 7	Interoperability with Other Systems	Ensure interoperability with other voting tracking or blockchain platforms. Support importing and exporting voting and metadata to and from other systems.
FR- 8	Content Preview and Playback	Provide the ability to preview or play voting directly within the system. Ensure compatibility with various file formats and viewers.
FR- 9	Reporting and Analytics	Generate reports on usage, ownership changes, and access patterns. Provide analytics to help users understand voting performance and trends.
FR- 10	Auditing and Compliance	Maintain an audit trail of all related activities. Ensure compliance with relevant data protection and copyright regulations.

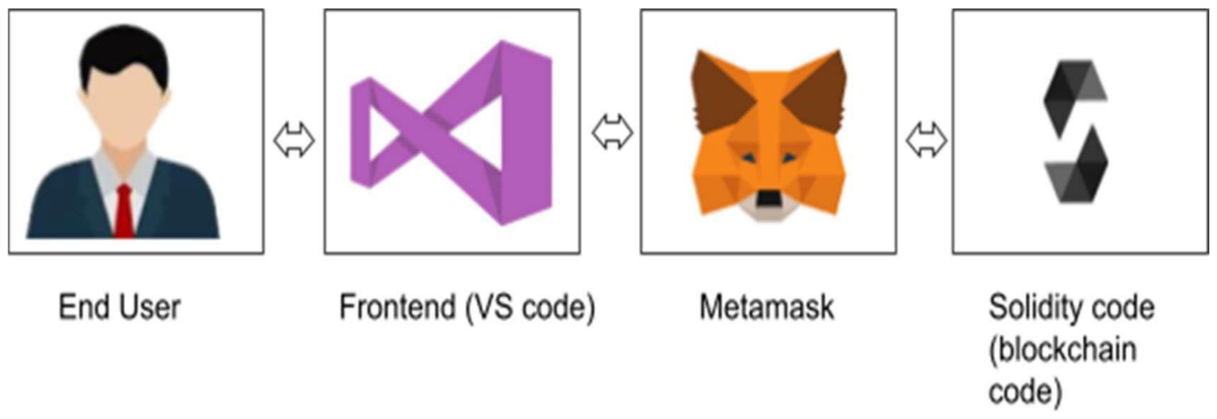
4.2 Non-Functional requirements

NFR No.	Non-Functional Requirement	Description

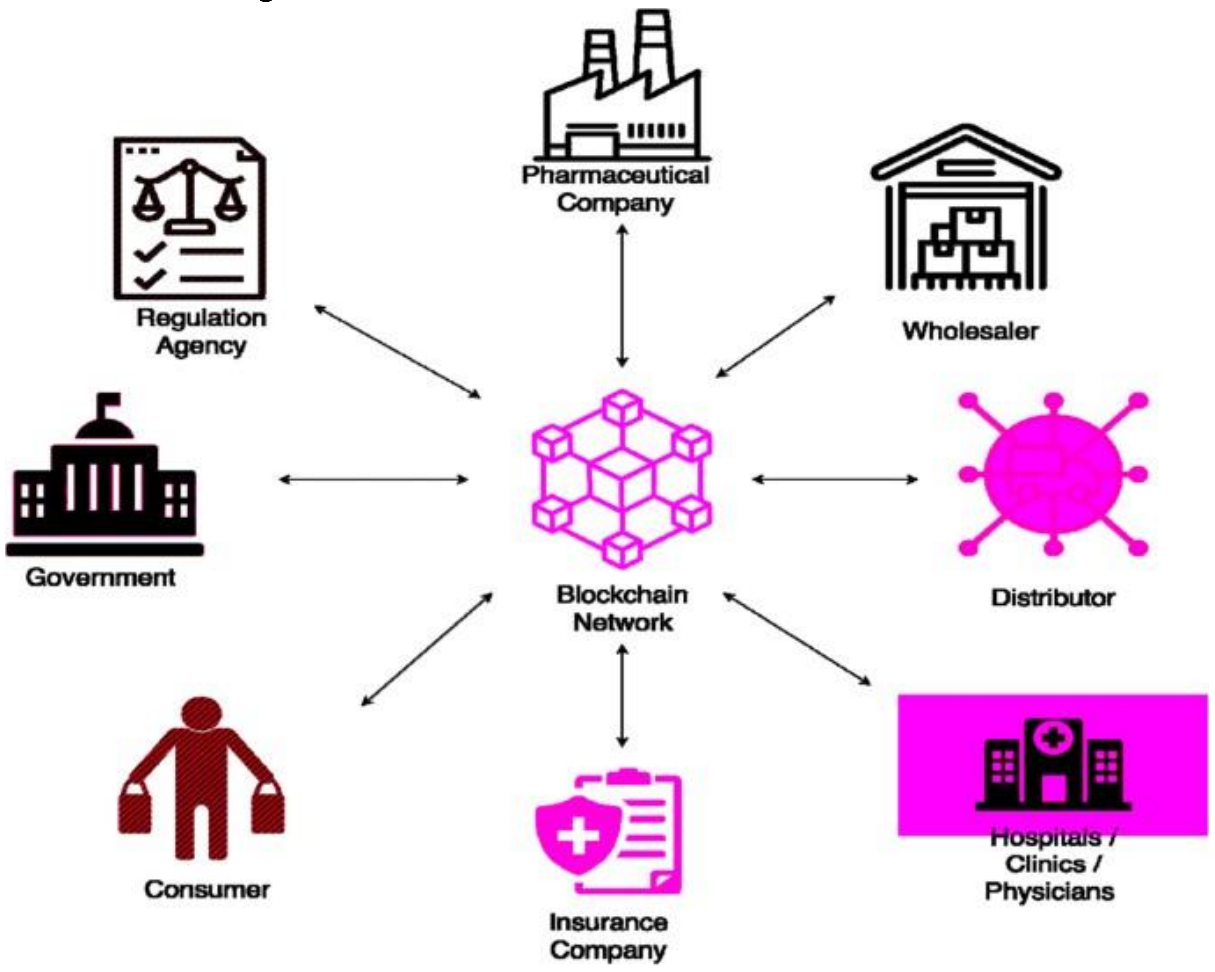
NFR-1	**Usability**	we've proposed a totally new blockchain system. This solution is more secure and scalable than other options currently available. Furthermore, the suggested system can prune its storage effectively, resulting in a robust and usable blockchain storage solution.
NFR-2	**Security**	Traditional solutions to achieve traceability within pharmaceutical supply chain are typically centralized and lack transparency across participants of the supply chain, which allows the central authority to modify information without notifying other stakeholders.
NFR-3	**Reliability**	Blockchain-based voting system offers a potential solution to create a distributed shared data platform for an immutable, trustworthy, accountable and transparent system in the PSC**.**
NFR-4	**Performance**	Scalability is essential, as the system must maintain usability as the volume of vote and users grows. Consistent response times are paramount, and well-defined benchmarks must be met to uphold usability standards. Performance testing and regular system optimization are essential to ensure that operates smoothly, regard less of the scale or usage patterns.
NFR-5	**Availability**	Blockchain technology ensures an efficient and cost-effective solution that underpins different functions and procedures to ascertain proper identification, tracing, tracking, and provenance.
NFR-6	**Scalability**	Blockchain technology enables creating a private permissioned network to trace and track events in the pharmaceutical supply chain and provides time stamped records of each transaction performed. Examples of events includes, execution and owner, time, location of transaction, and which stakeholders were involved.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories



Data Flow Diagrams

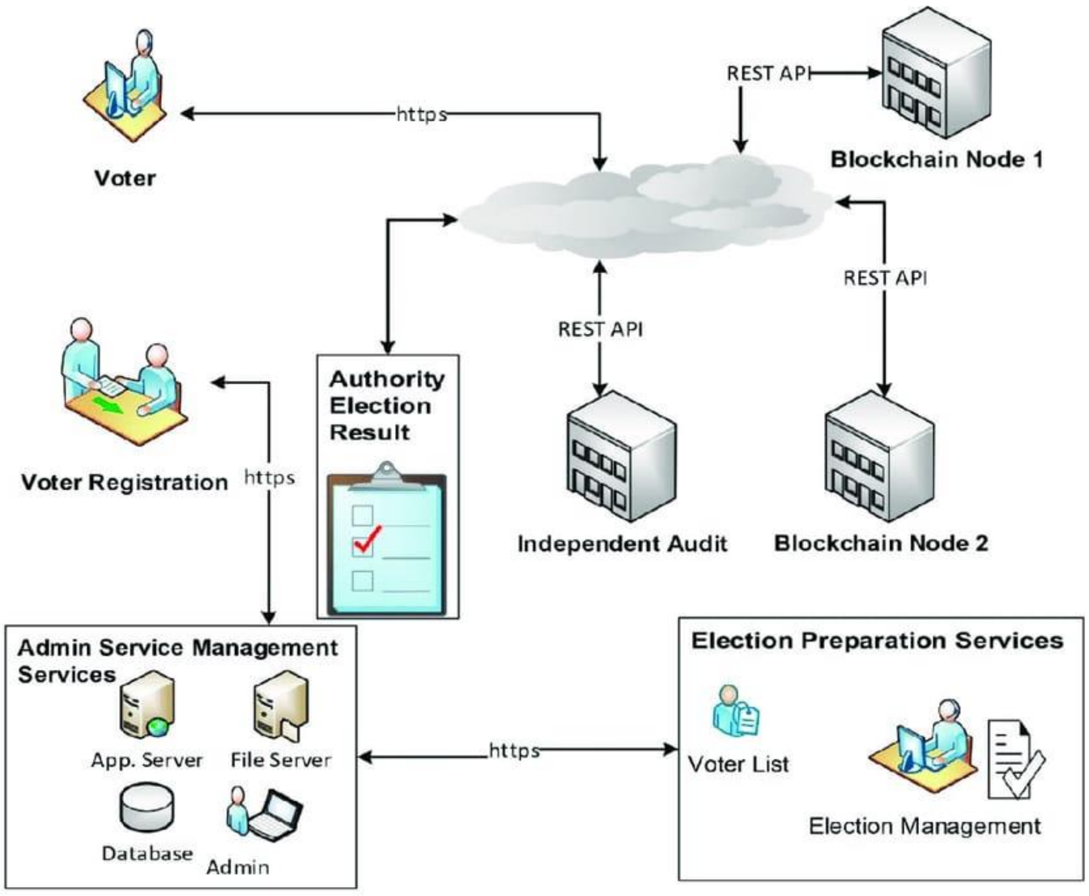


User Stories

User Type	Functional Requireme nt (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	team Member

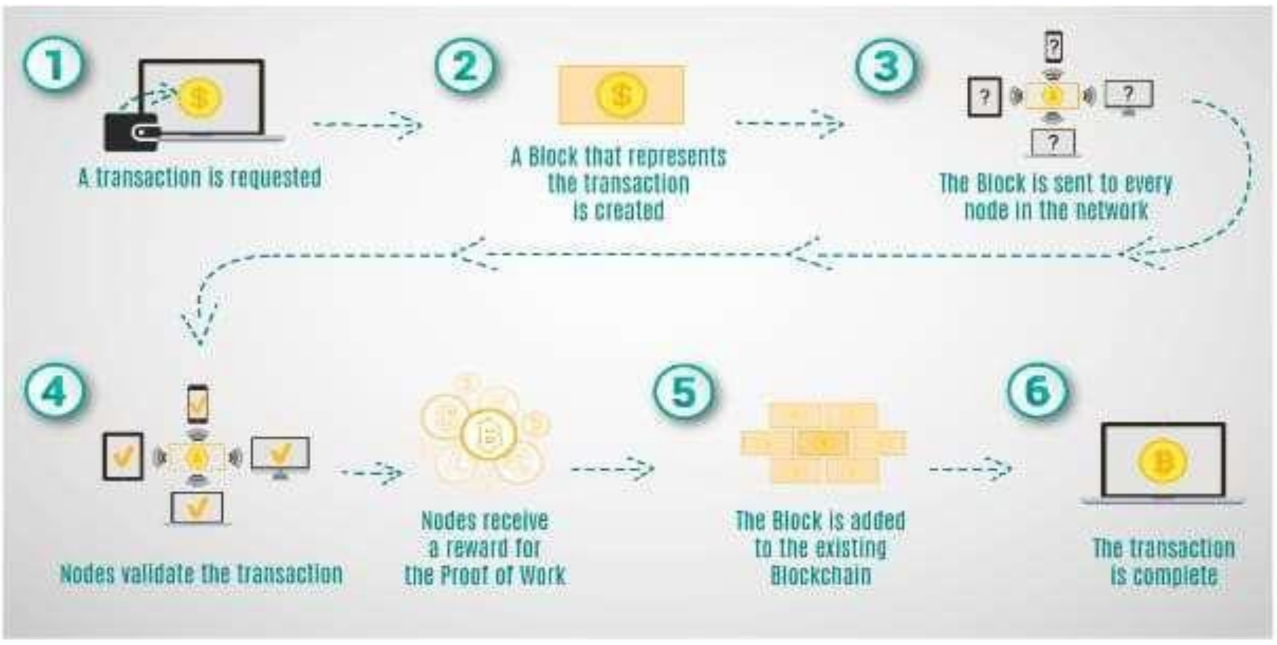
Content Creator	Vote Upload and Management	USN-1	As a content creator, I want to upload multiple image and video assets to the voting system with drag-and drop functionality	Users should be able to drag and drop multiple assets onto the voting system interface, and the system should process and upload them efficiently	High	Kaviyapriya sandhiya
Content Creator	vote Upload and Management	USN-2	As a content creator, I need to add detailed metadata to my vote, including titles, descriptions, and copyright information, to keep them well organized	Metadata fields should be easily accessible and editable, and changes should be immediately reflected in drug information	Medium	varshini
Marketing Manager	Vote Organization and Access Control	USN-3	As a marketing manager, I want to create and assign tags to vote for easy categorization, facilitating efficient retrieval.	Tags should be customizable, and vote should be sortable and filterable by assigned tags	Medium	Gopika kaviyapriya
Marketing Manager	Vote Organization and Access Control	USN-4	As a marketing manager, I need to restrict access to confidential to authorized team members only	Access control settings should allow me to specify who can view, edit, and delete drug , with permissions easily adjustable	High	Varshini sandhiya
Administrator	System Management	USN-5	As an administrator, I want to monitor and manage access, user roles, and system performance.	The admin dashboard should provide insights into user activity, allow role assignments, and offer system performance metrics	High	Gopika sandhiya
Administrator	System Management	USN-6	As an administrator, I need to set up automated data backups and a disaster recovery plan for data safety.discipline	The system should regularly back up data and provide a documented recovery plan to prevent data loss.	High	kaviyapriya

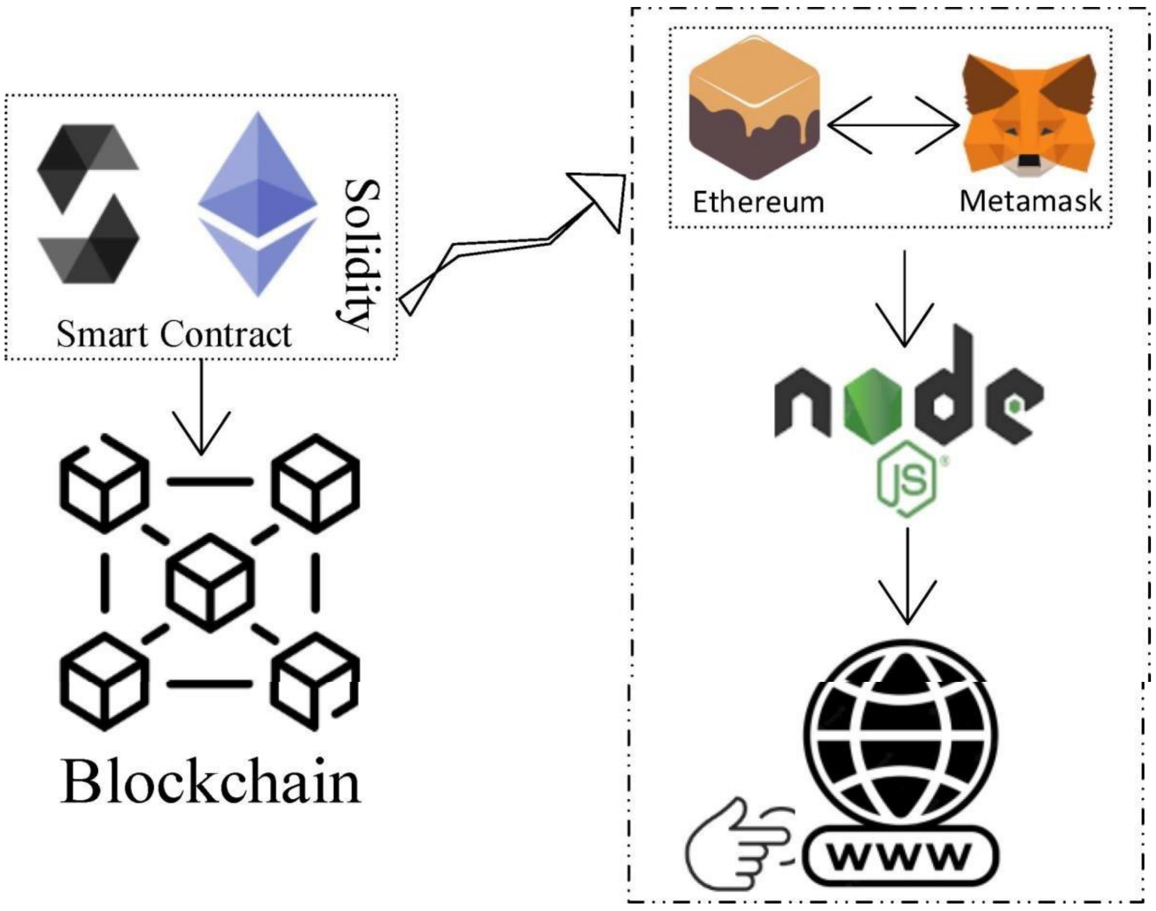
5.2 Solution Architecture



6.PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture


1
A transaction is requested
2
A Block that represents the transaction is created
3
The Block is sent to every node in the network
4
Nodes validate the transaction
Nodes receive a reward for the Proof of Work
5
The Block is added to the existing Blockchain
6
The transaction is complete



6.2 Sprint Planning & Estimation

1. User Story Backlog:

Start by creating a backlog of user stories. These stories should represent the features, enhancements, or tasks needed for your drug traceability. Ensure they are well-defined, with clear acceptance criteria.

2. Prioritization:

Collaborate with stakeholders to prioritize the user stories based on their importance and impact. High-priority stories should be at the top of the backlog.

3. Sprint Planning Meeting:

Hold a sprint planning meeting with your development team. During this meeting, select a set of user stories from the backlog to work on during the upcoming sprint. Consider the team's capacity and the complexity of the stories.

4. Story Point Estimation:

Use a method like story point estimation to estimate the effort required for each user story. The team assigns relative points to stories to indicate their complexity. This helps in determining how many stories can be included in the sprint.

5. Sprint Goal:

Define a clear sprint goal, which should align with the project's objectives. The goal should provide a sense of purpose for the sprint.

6. Daily Stand-Ups:

Conduct daily stand-up meetings to keep the team updated on progress, discuss any challenges, and make necessary adjustments to the sprint plan.

7. Sprint Review:

At the end of the sprint, hold a sprint review meeting to showcase the completed work to stakeholders. Gather their feedback and insights.

8. Sprint Retrospective:

After the review, conduct a sprint retrospective to assess what went well and what could be improved. Use this feedback to make process enhancements for the next sprint.

9. Continuous Improvement:

Agile principles emphasize continuous improvement. Apply lessons learned from each sprint to refine the process, including better estimation and planning.

10. Blockchain Integration Considerations:

When estimating and planning, consider the complexities related to blockchain integration, such as smart contract development, security measures, and the use of Ethereum's capabilities.

6.3 Sprint Delivery Schedule

1.Divide the Project into Sprints:

Begin by dividing the overall drug traceability project into sprints. Sprints are time-bound iterations, usually lasting 2-4 weeks, during which specific sets of features or tasks are completed.

2.Prioritize User Stories:

Review the prioritized user stories from your backlog and select those that will be addressed in each sprint. Ensure that each sprint has a clear focus and goal.

3.Define Sprint Durations:

Decide on the duration of each sprint. Agile sprints are typically 2-4 weeks long, but you can choose the duration that works best for your team and project.

3.Create a Sprint Backlog:

For each sprint, create a sprint backlog that includes the user stories, tasks, and features that will be tackled during that sprint.

4.Assign Story Points:

Estimate the effort required for each user story in the sprint backlog using story points or other estimation methods. This helps in understanding the capacity of the sprint.

5.Distribute Workload:

Based on the team's capacity and story point estimates, distribute the workload evenly across the sprint backlog items. Ensure that the team can realistically complete the planned work during the sprint.

6.Define Milestones:

Within each sprint, set specific milestones or checkpoints for key tasks or features. This helps in tracking progress and ensuring that the team is on target.

7.Adjust for Blockchain Integration:

Consider the complexities of blockchain integration in your delivery schedule. Tasks related to smart contract development, security testing, and Ethereumspecific considerations should be accounted for.

8.Iterative Development:

Remember that in Agile development, work is delivered incrementally. At the end of each sprint, you should have a potentially shippable product increment.

9.Continuous Review and Adaptation:

After each sprint, hold sprint reviews and retrospectives to gather feedback, evaluate progress, and make necessary adjustments to the delivery schedule or project priorities.

10.Release Planning:

Based on the progress in each sprint and the feedback received, plan releases of the drug traceability. These releases can be scheduled according to the completion of major features or project milestones.

11.Maintain a Release Calendar:

Maintain a release calendar that outlines when each sprint's deliverables or major releases are expected to be available to users or stakeholders. Share this calendar with the team and relevant parties.

7. CODING & SOLUTIONING (Explain the features added in the projectalong with code)

7.1 Feature 1

<u>Register vote</u>

- The "vote" struct represents the properties of a vote, including title, description, IPFS hash, and the owner's Ethereum address.
- The "vote" array stores registered drug.
- The " register vote" function allows a user to register a new vote by providing the title, description, and the IPFS hash of the drug data. It also records the user's Ethereum address as the owner.
- You can emit an event to log the vote registration, providing information about the newly registered vote.

Smart Contract (Solidity):

```
 // 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[_v
oterId].age,candidateMap[_voterId].voteCount);
 }

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

    return
(votersMap[_voterId].voterId,votersMap[_voterId].name,votersMap[_voterId].ag
e,votersMap[_voterId].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;
 }


}
```

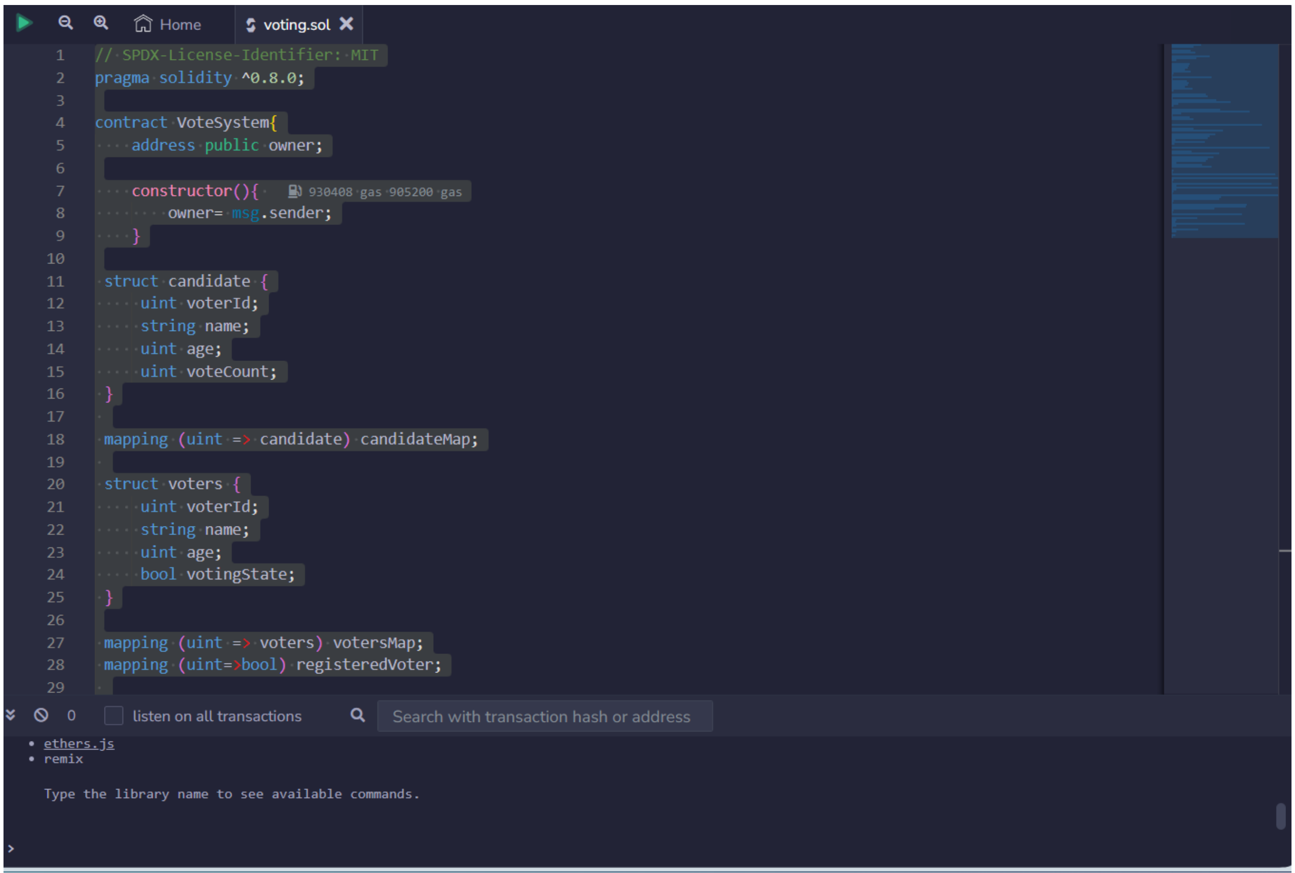
7.2 Feature 2

Transfer Ownership

1.The "vote" struct represents the properties of a vote, including title, description, IPFS hash, and the owner's Ethereum address.

- The "vote" array stores registered assets.
- The "register vote" function allows a user to register a new vote, which is owned by the user who registers it.
- The "transfer Ownership" function lets the owner of an vote transfer ownership to another user by specifying the drug's index and the address of the new owner.

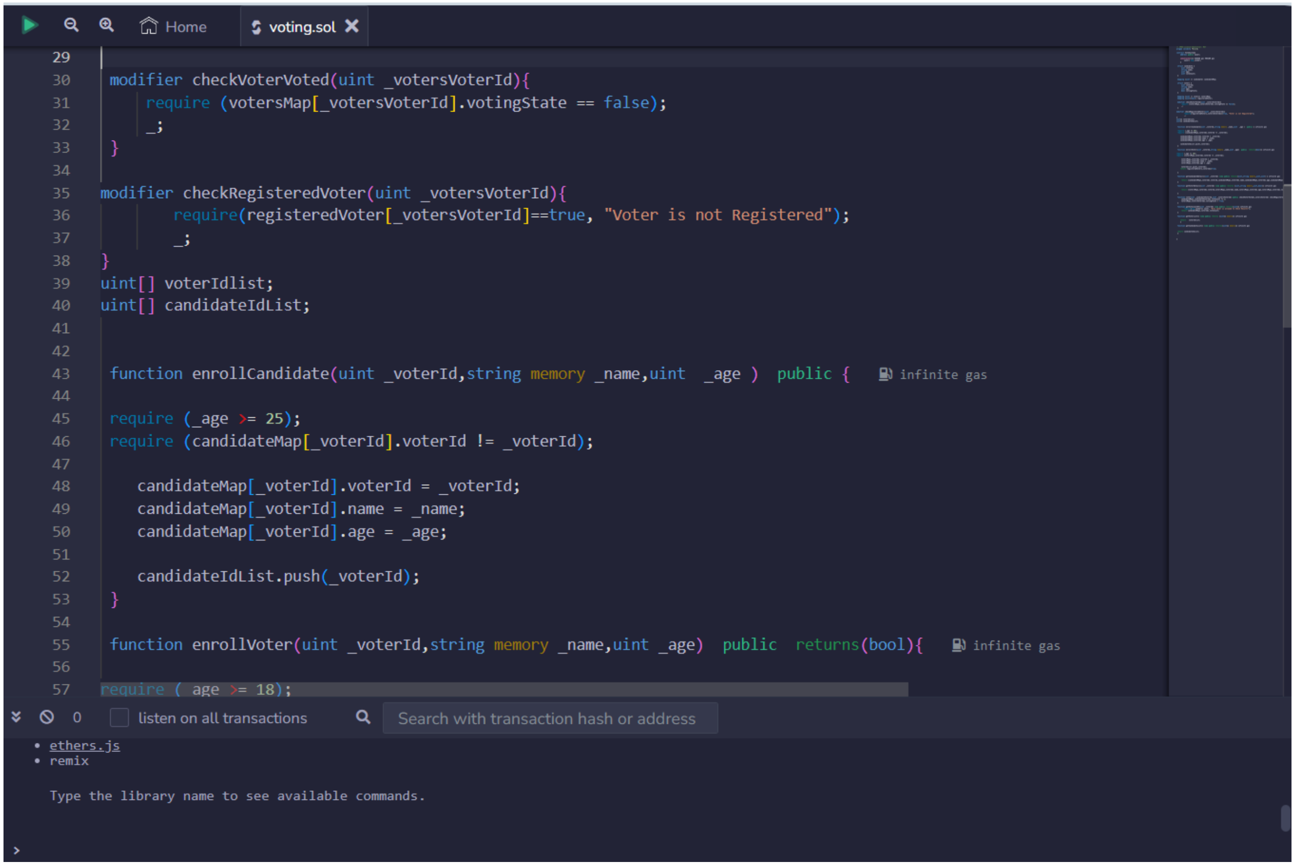
Smart Contract (Solidity):



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

contract VoteSystem{
    address public owner;

    constructor(){    930408 gas 905200 gas
        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;

```

0
listen on all transactions
Search with transaction hash or address

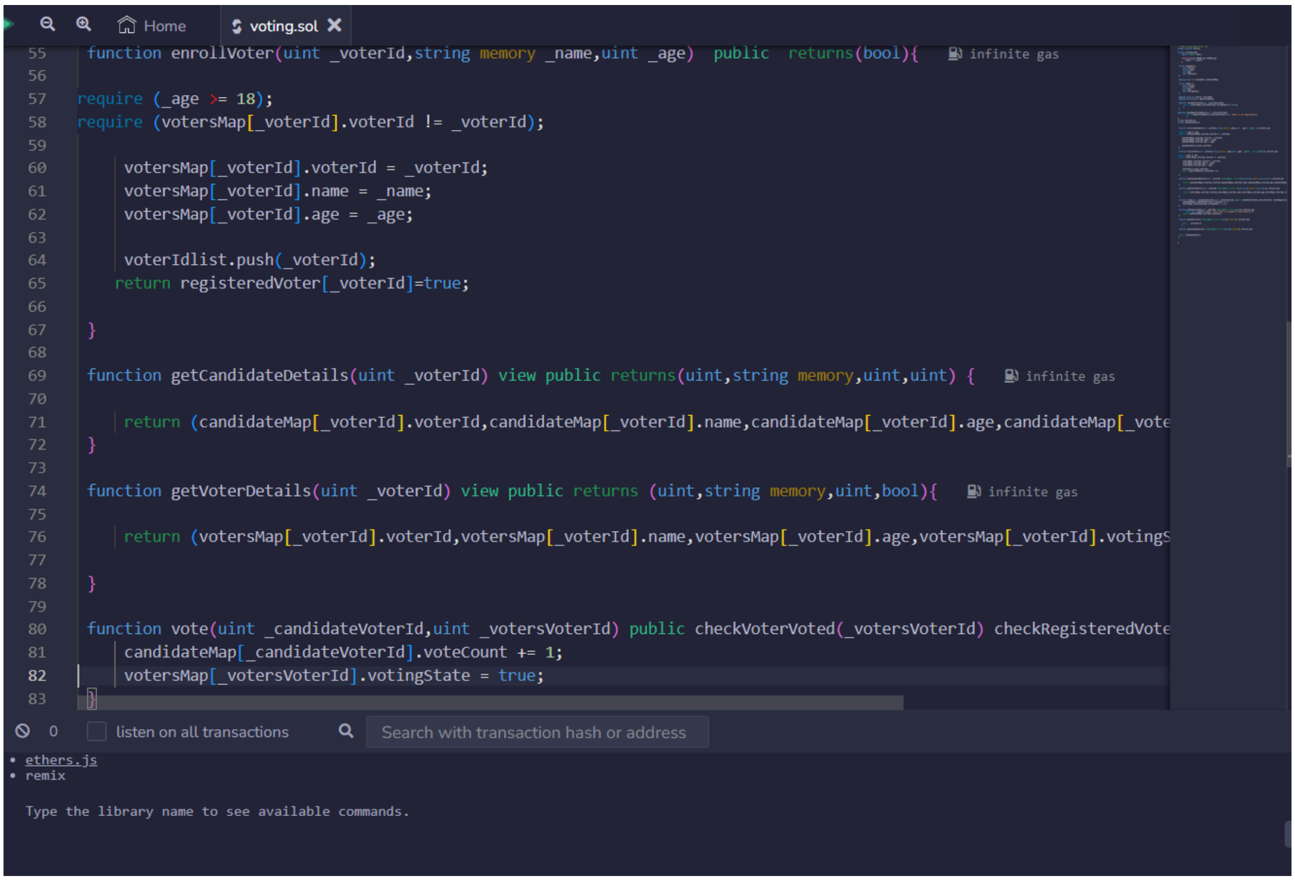
```
• ethers.js
• remix

  Type the library name to see available commands.

>
```



```

 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 {    infinite gas

 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){    infinite gas

require ( age >= 18);
```

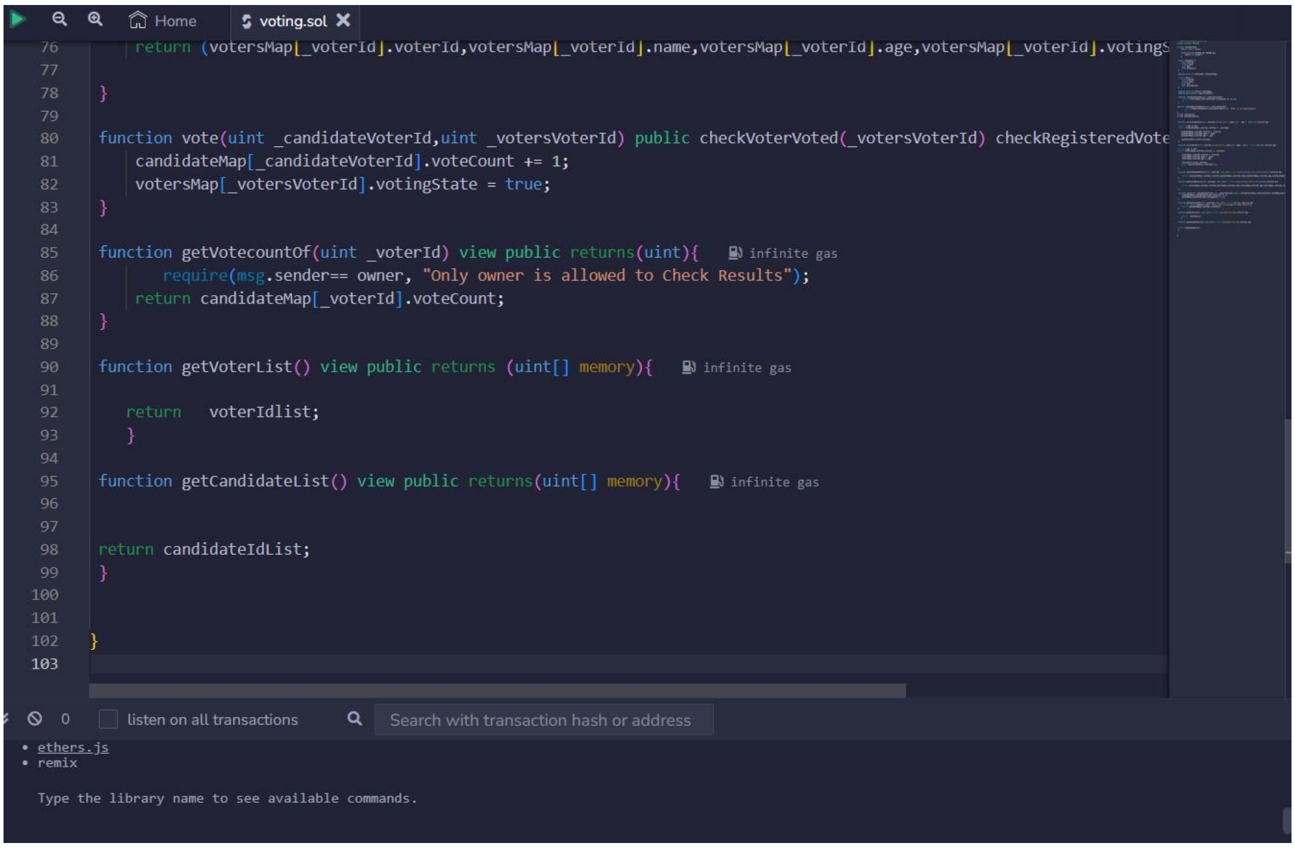
0
listen on all transactions
Search with transaction hash or address

```
• ethers.js
• remix

  Type the library name to see available commands.

>
```



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

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) {    infinite gas

    return (candidateMap[_voterId].voterId,candidateMap[_voterId].name,candidateMap[_voterId].age,candidateMap[_vote
}

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

    return (votersMap[_voterId].voterId,votersMap[_voterId].name,votersMap[_voterId].age,votersMap[_voterId].votingS

}

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

```
0   listen on all transactions   Search with transaction hash or address
• ethers.js
• remix

  Type the library name to see available commands.
```



```
    return (votersMap[_voterId].voterId,votersMap[_voterId].name,votersMap[_voterId].age,votersMap[_voterId].votingS

}

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

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

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

    return    voterIdlist;
    }

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


return candidateIdList;
}



}
```

```
0   listen on all transactions   Search with transaction hash or address
• ethers.js
• remix

  Type the library name to see available commands.
```

7.3 Database Schema (if Applicable)

The traditional database schema is replaced with smart contracts, which define the structure and behavior of vote and related data on the blockchain. However, certain off-chain data and metadata may be stored in traditional databases or decentralized storage systems for efficiency and scalability. Below, I'll provide a high-level overview of how the database schema for a electronic voting on the Ethereum blockchain might look:

On-Chain Ethereum Smart Contracts:

Vote Contract:

- Attributes:

drug title (string)
drug description (string) IPFS
hash for drug data (string)
Owner (address)
Public status (boolean)

- Functions:

Register drug
Transfer ownership
Toggle public status

Off-Chain Metadata Storage (Traditional Database or Decentralized Storage):

1.User Profiles:

- Attributes:

User ID
Ethereum address
Username
Email
Other user-specific details

2.Audit Trail:

- Attributes:

drug ID
Action
User performing the action
Timestamp
Details of the action

3.Asset Metadata:

- Attributes:

  drug ID

drug metadata

Additional details

This off-chain storage can include a traditional relational database for user profiles and audit trail data or decentralized storage systems like IPFS for storing vote metadata and potentially even the vote system files themselves.
It's important to note that the Ethereum blockchain is primarily used for storing critical asset ownership and transaction data, ensuring immutability and transparency. Off-chain storage is often used to handle less critical data and to optimize data access and retrieval times.
The database schema can vary significantly based on the specific requirements of the electronic voting system, and you may need to expand or modify the schema to suit your application's needs. The goal is to balance the benefits of blockchain's immutability with efficient data management and retrieval for users.

8. PERFORMANCE TESTING

8.1 Performace Metrics

Asset Upload and Retrieval Speed:

Metric: Average time taken to upload and retrieve vote.
Importance: Measures the speed of electronic voting system ensuring quick access to vote.

Blockchain Transaction Throughput:

Metric: Transactions per second (TPS) on the Ethereum blockchain.
Importance: Indicates how well the system handles blockchain transactions, which is crucial for scalability.

Smart Contract Execution Time:

Metric: Average time taken for smart contract execution. Importance: Evaluates the efficiency of the blockchain-based logic governing vote ownership and access.

Vote Metadata Search Time:

Metric: Time it takes to search for vote based on metadata. Importance: Measures the responsiveness of the system's search functionality.

User Authorization Latency:

Metric: Time it takes to validate and authorize user access to vote.
Importance: Ensures that authorized users can access vote promptly while maintaining security.

Storage Space Usage:

Metric: Amount of blockchain storage used by vote and associated data.
Importance: Evaluates the cost and efficiency of storage on the blockchain.

vote Accessibility Uptime:

Metric: Percentage of time vote are accessible.
Importance: Measures the system's reliability and availability for users.
Security Audit Findings:

Metric: Number and severity of security vulnerabilities discovered during audits.
Importance: Identifies potential risks and the need for security improvements.

User Feedback and Satisfaction:

Metric: User surveys or feedback on system usability and performance.
Importance: Provides insights into user satisfaction and areas for improvement.

Ethereum Network Gas Costs:

Metric: Total gas costs incurred for transactions and contract interactions.
Importance: Measures the cost-efficiency of system operations.

Scalability Metrics:

Metric: System's ability to handle an increasing number of users and vote.

Importance: Assesses how well the system can scale with growing demands.

Audit Trail Accuracy:

Metric: Accuracy and completeness of the audit trail for voting system
Importance: Ensures a reliable record of vote history for compliance and accountability.

Data Backup and Recovery Time:

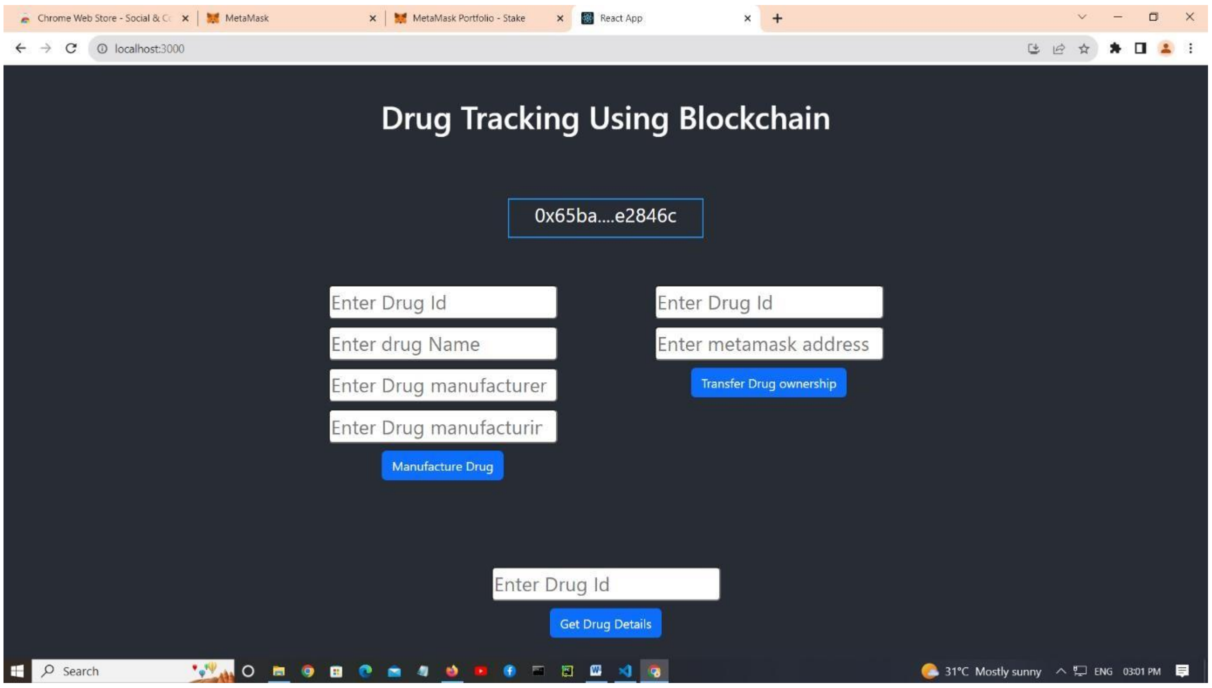
Metric: Time taken for data backup and recovery operations.
Importance: Evaluates the system's readiness for data recovery in case of failures.

9. RESULTS

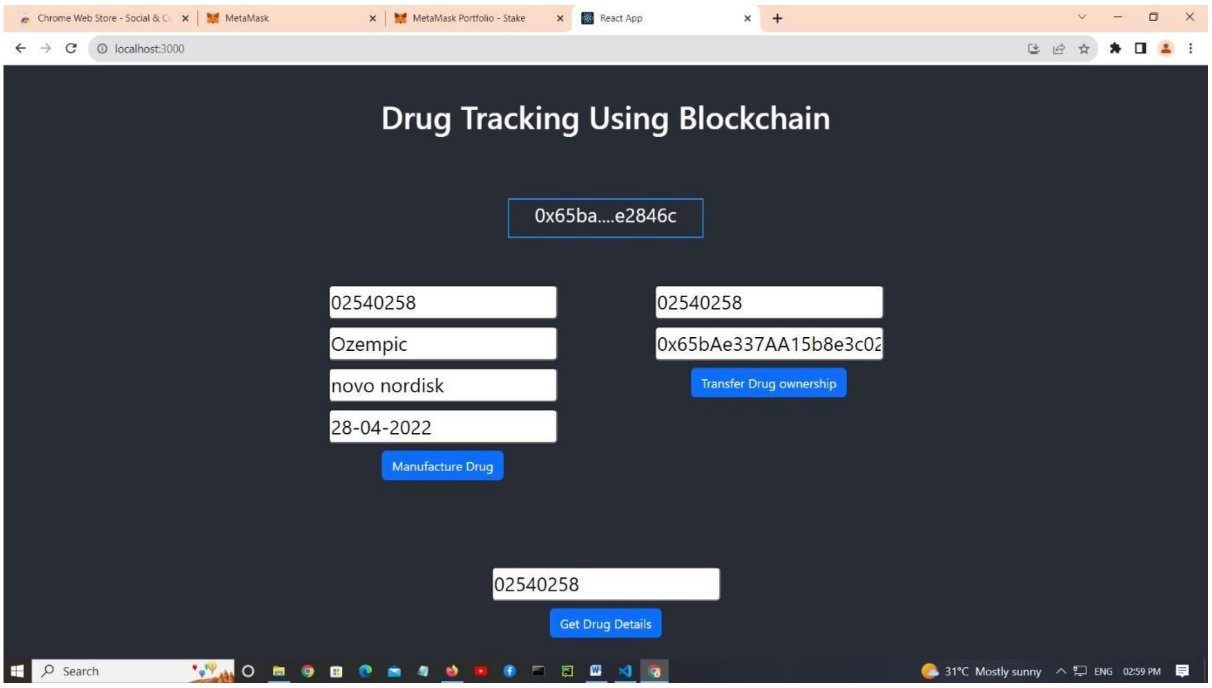
9.1 Output Screenshots

The " vote" feature allows users to officially record and store information about the electronic voting on the Ethereum blockchain. This process establishes proof of ownership and a public record of the vote’s existence. It is marked as "public" and can be viewed or accessed by anyone on the Ethereum blockchain.

Get drug tracking using blockchain is showing a status of the drug (manufacturing vote, transfer vote ownership)



"Transfer ownership" in the context of a voting system on the Ethereum blockchain refers to the ability of the current owner of a vote details to transfer ownership rights to another user. This feature allows drug owners to change the entity or individual that has control and ownership of a particular vote.



Drug Tracking Using Blockchain

0x65ba....e2846c

02540258
Ozempic
novo nordisk
28-04-2022
Manufacture Drug

02540258
0x65bAe337AA15b8e3c02
Transfer Drug ownership

02540258
Get Drug Details

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Immutability and Transparency: All vote transactions and ownership changes are recorded on the Ethereum blockchain, providing an immutable and transparent audit trail. This enhances trust and security.

- Ownership Verification: Ethereum's smart contracts allow for clear ownership verification, reducing disputes and proving the authenticity of vote.

- Decentralization: The Ethereum blockchain operates on a decentralized network, reducing the risk of single points of failure and enhancing security and availability.

- Security: vote stored on the blockchain benefit from robust cryptographic security, making it difficult for unauthorized parties to tamper with or steal vote.

- Global Accessibility: Ethereum is a global network, making vote accessible to a worldwide audience without geographic restrictions.

- Interoperability: Ethereum's compatibility with various standards and protocols allow for easy integration with other blockchain-based systems.

- Cost-Effective: Smart contracts can automate voting system processes, reducing the need for intermediaries and saving costs.

DISADVANTAGES

- Scalability: Ethereum has faced challenges related to network congestion and scalability, making it less suitable for high-frequency vote system.

- Gas Costs: Every operation on the Ethereum blockchain consumes gas
- (transaction fees), which can make frequent drug tracking expensive.

- User Experience: Interacting with the blockchain can be complex for nontechnicalusers, leading to usability challenges.

- Regulatory Concerns: Blockchain-based vote may raise regulatory questions, particularly if they represent real-world vote or securities.

- Data Storage: Storing large vote directly on the blockchain can be inefficient and costly. Many vote are stored off-chain or on decentralized storage systems like IPFS.

- Irreversible Transactions: Once a transaction is confirmed on the Ethereum blockchain, it is irreversible. Mistakes can be costly.
Smart Contract Vulnerabilities: Poorly written smart contracts can lead to security vulnerabilities and hacks, resulting in loss of drug.

- Privacy: The Ethereum blockchain is public, which means that drug data is visible to anyone. For private vote, additional privacy measures are needed.

11. CONCLUSION

The integration of electronic voting systems with blockchain technology offers significant potential to enhance the integrity, security, and transparency of elections. By leveraging the decentralized nature of blockchain, it becomes possible to create a tamper-proof and immutable ledger of votes. This ensures that once a vote is cast, it cannot be altered or manipulated, thereby increasing the overall trust in the electoral process.

Additionally, blockchain-based electronic voting systems can enhance accessibility, allowing citizens to vote from the convenience of their homes using secure digital identities. This inclusivity fosters higher voter turnout and engagement.

12.FUTURE SCOPE

1. Develop a prototype to demonstrate the requirements shared via the guidelines of the DGFT/DAVA Team. Learned how the GS1 enabled 2D/1D barcodes will be associated with each drug pack and document the shortcoming in any.

2. Involved an early adopter to participate in our Traceability CRM Solution. We had two early adopters from the Pharma Industry based out in Kothur, Hyderabad, and another firm based out at Pashamylaram, Hyderabad. The former allowed us to gather requirements at the factory and allowed us to work with production packaging /warehouse teams.

3. I began with the bottom-up approach while documenting the requirements. As per guidelines from FDA (Implementation in a hospital pharmacy in Argentina" GS1.org, 2014) , (GS1 Standards in the Pharmaceutical Supply Chain, 2018), from MHRA (GOV.UK, 2021) , From GS1 (Traceability system a must for drugs: GS1 chief , 2021) Based on the expectations, we learned from three guidelines from DGFT-DAVA/US FDA-US DSCSA /MHRA-FMD (Falsified Medical Directives).

4. Understanding the component of Serialization requirement. As per the initial user specification document composed by SolutionsMax Technology Services (An Enterprise Solutions for the Pharma Ecosystem, 2016-2021) Organizations need to identify various components such as the readiness of the ERP system as a master data repository, changes in artwork for all affected Stock Keeping Units (SKUs), and new systems such as the enterprise serialization manager, packaging line system, and edge systems.

5. Types of Business Reports measuring compliance attributes. . As per the initial user specification document composed by SolutionsMax Technology Services (An Enterprise Solutions for the Pharma Ecosystem, 2016-2021)Emphasis was to understand the various quality and business reporting that would be required to beestablished. Establishing a parent-child relationship between the packs that are being packaged so that traceability can be reported

13.APPENDIX Source Code

Solidity coding :

```
// SPDX-License-Identifier: MIT
pragmasolidity^0.8.0;
 contract
Drug{
    addresspublic owner;

constructor(){
        owner =msg.sender;
    }      modifier onlyOwner(){          require(msg.sender == owner,"Only
the owner can perform this action");
        _;
    }      struct Drug {
string drugName;         string
manufacturer;         uint256
manufacturingDate;
address trackingHistory;
    }      mapping(uint256 =>
Drug)public drugs;     uint256public
drugCount;

    event DrugManufactured(uint256indexed drugId,string
drugName,string manufacturer,uint256 manufacturingDate);      event
DrugTransferred(uint256indexed drugId,addressindexed
from,addressindexed to,uint256 transferDate);

    function manufactureDrug(uint256 drugId,stringmemory
_drugName,stringmemory _manufacturer,uint256
_manufacturingDate)external onlyOwner {
        address initialHistory;
initialHistory = owner;
```

```
        drugs[drugId]= Drug(_drugName, _manufacturer, _manufacturingDate,
initialHistory);         drugCount++;
         emit DrugManufactured(drugId, _drugName, _manufacturer,
_manufacturingDate);
    }     function transferDrugOwnership(uint256 _drugId,address _to)external{
require(_to  !=address(0),"Invalid  address");                require(_to  !=
drugs[_drugId].trackingHistory,"Already owned by the new address");
         address from = drugs[_drugId].trackingHistory;
drugs[_drugId].trackingHistory = _to;
         emit DrugTransferred(_drugId, from, _to,block.timestamp);
}      function getDrugDetails(uint256
_drugId)externalviewreturns(stringmemory,stringmemory,uint256,address){
        Drug memory drug = drugs[_drugId];         return(drug.drugName,
drug.manufacturer, drug.manufacturingDate, drug.trackingHistory);
    }
}
```

Java script :

```
const { ethers } = require("ethers");
 const abi =
[  {
  "inputs": [],
  "stateMutability": "nonpayable",
  "type": "constructor"
 },
 {
  "anonymous": false,
  "inputs": [
   {
    "indexed": true,
    "internalType": "uint256",
    "name": "drugId",
    "type": "uint256"
   },
   {
      "indexed": false,
      "internalType": "string",
      "name": "drugName",
      "type": "string"
     },
     {
      "indexed": false,
      "internalType": "string",
      "name": "manufacturer",
      "type": "string"
     },
     {
      "indexed": false,
      "internalType": "uint256",
      "name": "manufacturingDate",
      "type": "uint256"
     }
    ],
    "name": "DrugManufactured",
    "type": "event"
  },
  {
    "anonymous": false,
    "inputs": [
     {
      "indexed": true,
      "internalType": "uint256",
      "name": "drugId",
      "type": "uint256"
     },
     {
      "indexed": true,
      "internalType": "address",
      "name": "from",
      "type": "address"
     },
     {
      "indexed": true,
      "internalType": "address",
      "name": "to",
      "type": "address"
     },
     {
      "indexed": false,
      "internalType": "uint256",
        "name": "transferDate",
      "type": "uint256"
     }
    ],
    "name": "DrugTransferred",
 "type": "event"
},
{
 "inputs": [],
 "name": "drugCount",
 "outputs": [
  {
   "internalType": "uint256",
   "name": "",
   "type": "uint256"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [
  {
   "internalType": "uint256",
   "name": "",
   "type": "uint256"
  }
 ],
 "name": "drugs",
 "outputs": [
  {
   "internalType": "string",
   "name": "drugName",
   "type": "string"
  },
  {
   "internalType": "string",
   "name": "manufacturer",
   "type": "string"
  },
  {
   "internalType": "uint256",
   "name": "manufacturingDate",
   "type": "uint256"
  },
  {
   "internalType": "address",
   "name": "trackingHistory",
        "type": "address"
  } ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [
    {
     "internalType": "uint256",
     "name": "_drugId",
     "type": "uint256"
    }
   ],
   "name": "getDrugDetails",
   "outputs": [
    {
     "internalType": "string",
     "name": "",
     "type": "string"
    },
    {
     "internalType": "string",
     "name": "",
     "type": "string"
    },
    {
     "internalType": "uint256",
     "name": "",
     "type": "uint256"
    },
    {
     "internalType": "address",
     "name": "",
     "type": "address"
    }
   ],
   "stateMutability": "view",
   "type": "function"
  },
  {
   "inputs": [
    {
     "internalType": "uint256",
     "name": "drugId",
     "type": "uint256"
    },
    {
     "internalType": "string",
     "name": "_drugName",
     "type": "string"
    },
    {
     "internalType": "string",
     "name": "_manufacturer",
     "type": "string"
    },
    {
```

```
    "internalType": "uint256",
    "name": "_manufacturingDate",
    "type": "uint256"
   }
  ],
  "name": "manufactureDrug",
  "outputs": [],
  "stateMutability": "nonpayable",
  "type": "function"
 },
 {
  "inputs": [],
  "name": "owner",
  "outputs": [
   {
    "internalType": "address",
    "name": "",
    "type": "address"
   }
  ],
  "stateMutability": "view",
  "type": "function"
 },
 {
  "inputs": [
   {
    "internalType": "uint256",
    "name": "_drugId",
    "type": "uint256"
   },
   {
    "internalType": "address",
    "name": "_to",
    "type": "address"
   }
  ],
  "name": "transferDrugOwnership",
  "outputs": [],
  "stateMutability": "nonpayable",
  "type": "function"
 }
]  if
(!window.ethereum)
{  alert('Meta Mask Not Found')
window.open("https://metamask.io/download/")
}  export const provider = new
ethers.providers.Web3Provider(window.ethereum); export const signer =
provider.getSigner(); export const address =
"0x65bAe337AA15b8e3c028B79E925FdB7eb9E2846c"
```

```
 export const contract = new ethers.Contract(address, abi,
signer)
```

HTML coding:

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta      name="description"      content="Web
site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
-->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

GitHub :

https://github.com/EVuqBqhz/Nan-mudhalvan-

Project Video demo link:

https://github.com/EVuqBqhz/Nan-mudhalvan-