BIOMETRIC SECURITY SYSTEM FOR VOTING PLATFORM

Bharath Yuvaneshwaran Surenthar Jai Surya

KCG COLLEGE OF TECHNOLOGY
CHENNAI - 600097

Table of Content

1.INTRODUCTION

1.1 Project Overview

In an era where secure and transparent voting systems are of utmost importance, our project aims to create a revolutionary voting platform that harnesses the power of biometric authentication and blockchain technology to ensure the utmost integrity and security of the voting process. This innovative platform is set to transform the way elections are conducted, offering a tamper-proof, anonymous, and user-friendly voting experience for citizens.The core components of this platform are designed to address the most pressing issues in the current electoral systems. Biometric authentication serves as the foundation, guaranteeing that only eligible voters can cast their ballots, significantly reducing the risk of identity fraud or unauthorized voting. This may involve fingerprint or iris scanning, facial recognition, or other advanced biometric methods.The integration of blockchain technology is pivotal in ensuring transparency and immutability throughout the entire voting process. Each vote is recorded as a unique transaction on the blockchain, making it virtually impossible to alter or delete votes once they are cast. The decentralized nature of the blockchain further safeguards the system from manipulation by any central authority.To make this platform accessible and user-friendly, we'll provide a secure mobile application that guides users through the biometric authentication process and voting options. Citizens can cast their votes from the comfort of their homes or designated polling stations, ensuring voting accessibility for all.Privacy and anonymity are paramount in our design. The blockchain ledger will only store encrypted data, thereby protecting the identity of the voter. Each voter will be assigned a unique, anonymous identifier, preventing any breach of their personal information.Transparency is a fundamental principle of this project. The blockchain ledger will be publicly accessible, enabling anyone to audit the voting results while maintaining voter anonymity. This openness is crucial in instilling trust in the electoral process and promoting a sense of accountability.

1.2 **Purpose**

The primary purpose of our project is to revolutionize the way elections are conducted by creating a secure, transparent, and user-friendly voting platform. We aim to enhance the security and integrity of the voting process through the implementation of cutting-edge biometric authentication methods, such as fingerprint or iris scanning and facial recognition. By leveraging blockchain technology, we ensure transparency and immutability in the voting process, making it virtually impossible to tamper with or manipulate voting results. Our project is driven by a commitment to safeguard the privacy and anonymity of voters. Through the use of encrypted data and unique, anonymous identifiers, we prioritize the protection of individual voter information. The development of a user-friendly mobile app facilitates convenient and accessible voting from various locations, ensuring that the electoral process is inclusive and efficient.The adoption of a decentralized network for blockchain maintenance further enhances security, making the system highly resistant to hacking attempts. Once a vote is cast, it becomes an immutable record on the blockchain, allowing for easy auditing and verification of voting results

2. LITERATURE SURVEY

2.EXISTING PROBLEM

2.1 Existing Problem

The existing problem with traditional voting systems is the lack of robust security and transparency. Biometric security has the potential to enhance the authentication process, but there are still concerns about its accuracy and privacy. Additionally, current voting systems often lack transparency and can be susceptible to fraud or manipulation. Blockchain technology, with its decentralized and

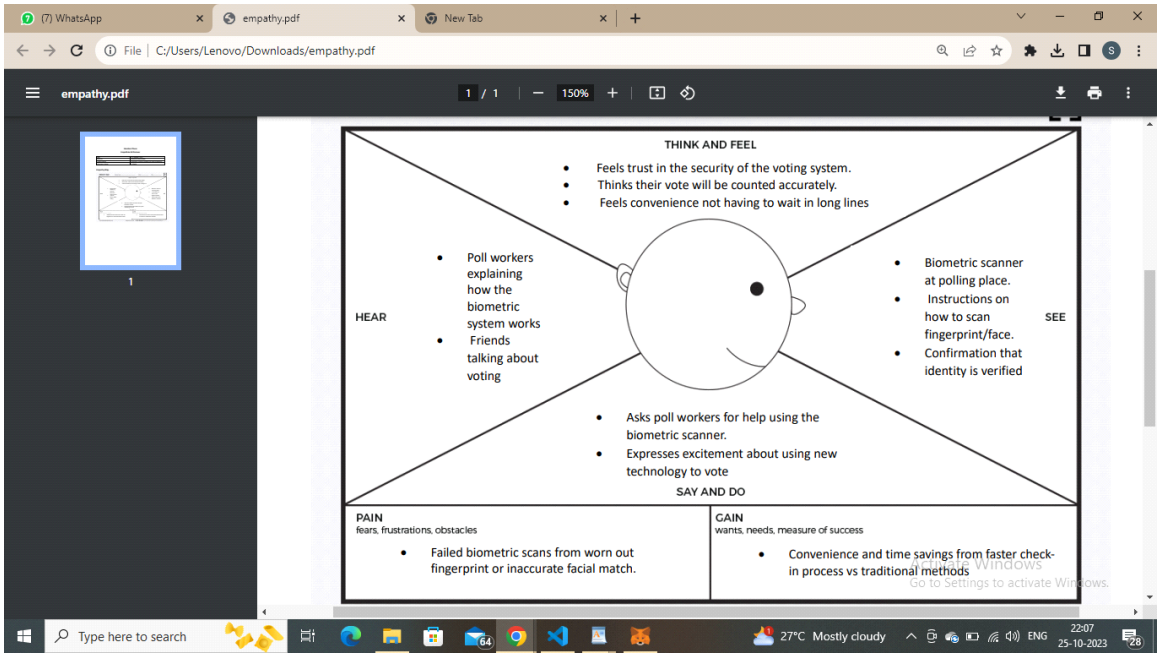
tamper-resistant ledger, offers a solution, but its integration into voting platforms is challenging and requires addressing scalability and accessibility issues. A successful biometric security system for voting on a blockchain would need to overcome these obstacles to ensure secure and trustworthy elections.The existing problem with traditional voting systems is the lack of robust security and transparency. Biometric security has the potential to enhance the authentication process, but there are still concerns about its accuracy and privacy. Additionally, current voting systems often lack transparency and can be susceptible to fraud or manipulation. Blockchain technology, with its decentralized and tamper-resistant ledger, offers a solution, but its integration into voting platforms is challenging and requires addressing scalability and accessibility issues. A successful biometric security system for voting on a blockchain would need to overcome these obstacles to ensure secure and trustworthy elections.

2.2 Problem Statement Definition

introducing biometric security systems to voting can improve identity verification, as biometric data like fingerprints or iris scans is more difficult to forge. However, implementing biometrics on a large scale poses challenges. Biometric data privacy is a major concern, and any breach of this sensitive information could have severe consequences. Additionally, ensuring the accuracy and consistency of biometric authentication across a diverse population is complex.

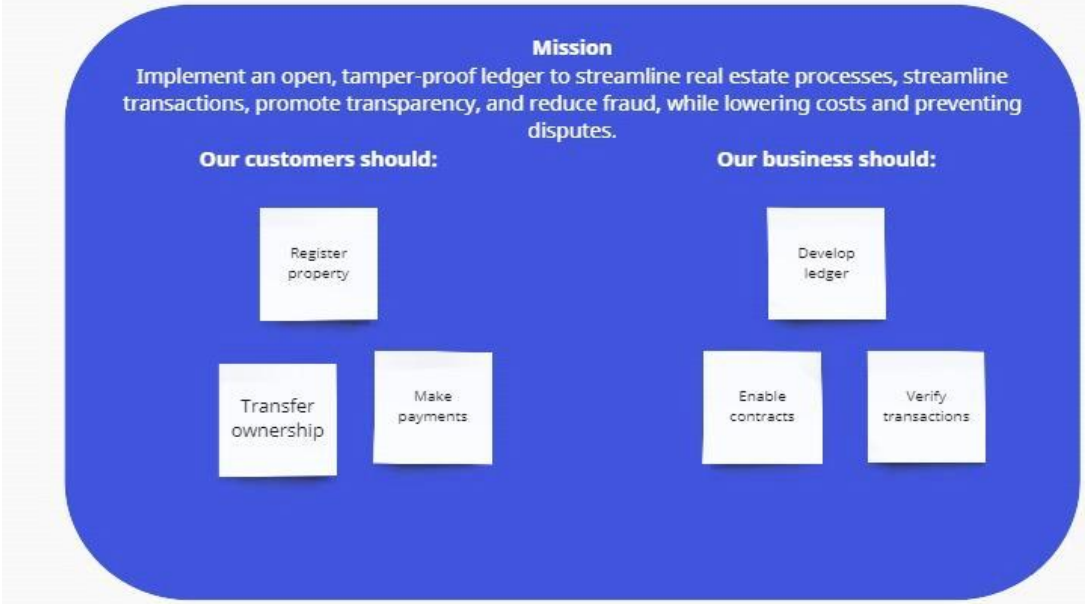
3.IDEATION AND PROPOSED SOLUTION

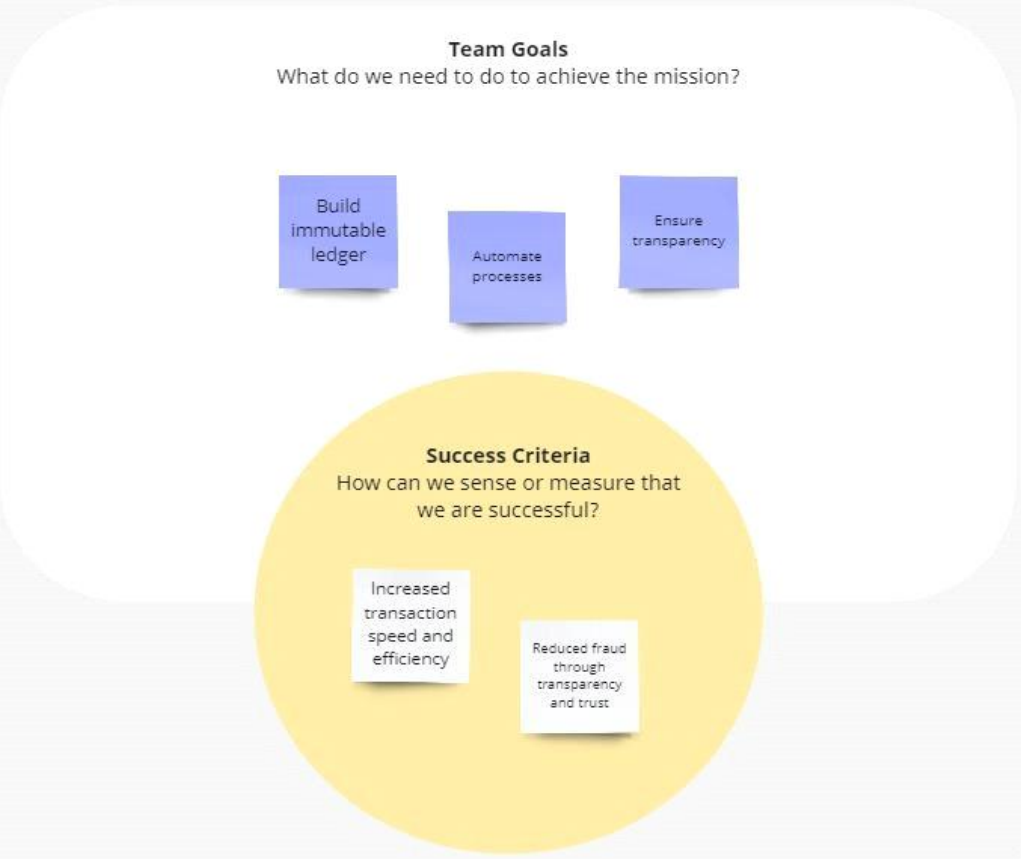
3.1 Empathy Map Canvas

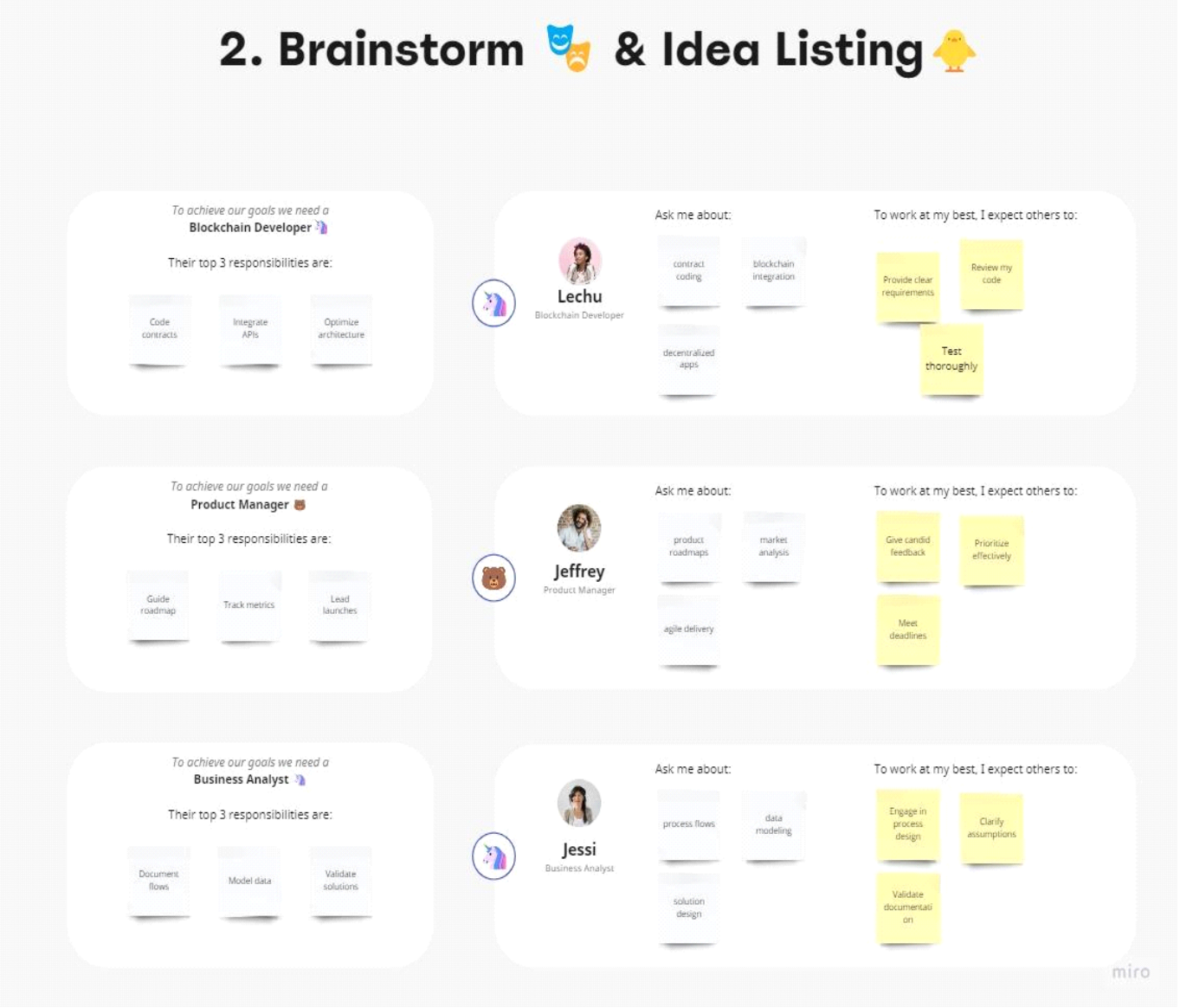


3.2 Ideation and Brainstroming

1. Team Gathering 🎯







4.REQUIREMENT ANALYSIS

4.1 Functional Requirement

4.1 Functional Requirements:

- User Registration: The system should allow voters to register their biometric data securely and efficiently.
- Biometric Authentication: The system should authenticate voters' biometric data during the voting process to ensure the accuracy and uniqueness of voter identification.
- Vote Recording: The system should record and store votes securely on the blockchain, ensuring transparency and immutability.
- Vote Counting: The system should accurately count the votes recorded on the blockchain and generate the final results.

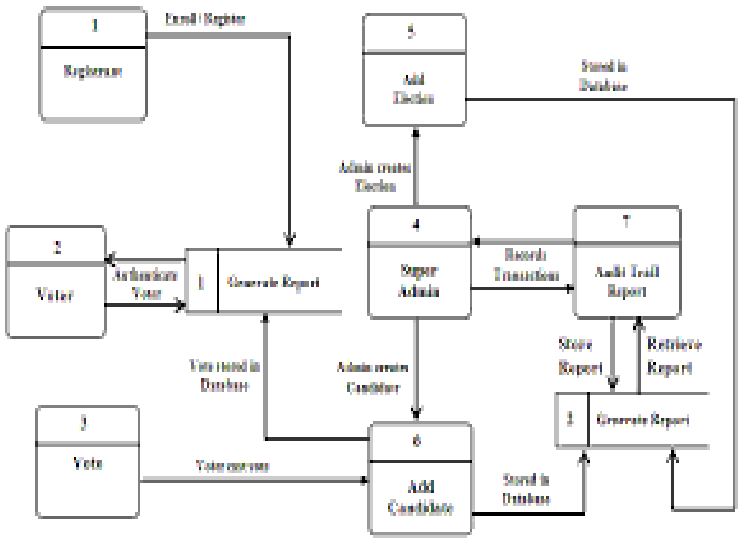
- Audit Trail: The system should provide a transparent audit trail, allowing authorized individuals or organizations to verify the integrity of the voting process.

4.2 Non-Functional Requirements:

- Security: The system should implement robust encryption and data protection measures to ensure the privacy and security of voters' biometric data and voting records.
- Scalability: The system should be able to handle a large number of voters and votes without compromising performance.
- Usability: The system should be user-friendly, intuitive, and accessible to all voters, including those with disabilities.
- Reliability: The system should be highly reliable, ensuring the availability and integrity of the voting process.
- Compliance: The system should adhere to relevant legal and regulatory requirements, such as data protection and privacy laws.

5.PROJECT DESIGN

5.1 Data Flow Diagram and User Stories



Story 1

As a user, I want to create a decentralized identity on the Ethereum blockchain so that I can have full control over my personal information and digital assets. want to link my Ethereum wallet address to my decentralized identity so that I can use it for authentication and authorization purposes. I want to store my personal information (such as name, email, and other relevant details) securely within my decentralized identity smart contract.

Story 2

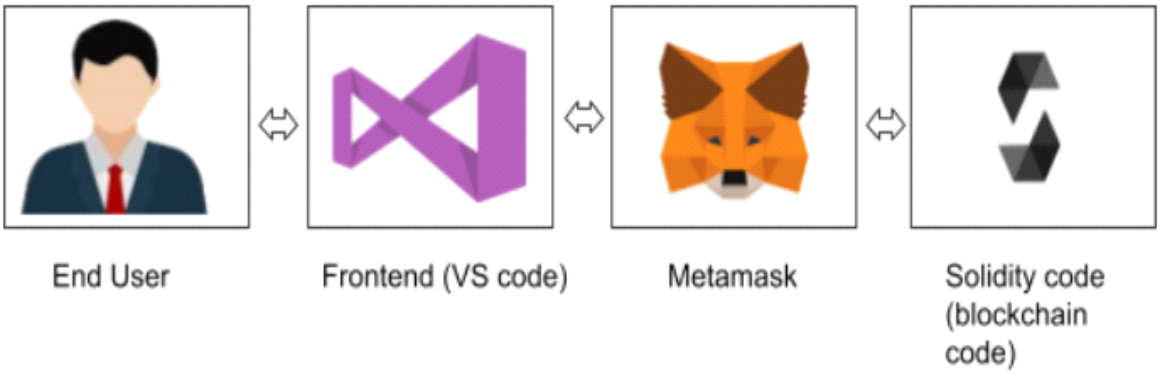
As a user, I want to share specific elements of my decentralized identity, such as my email address, with a third party in a secure and privacy-preserving manner.As a user, I want to have the ability to update and revoke access to my personal information from third parties I've shared it with

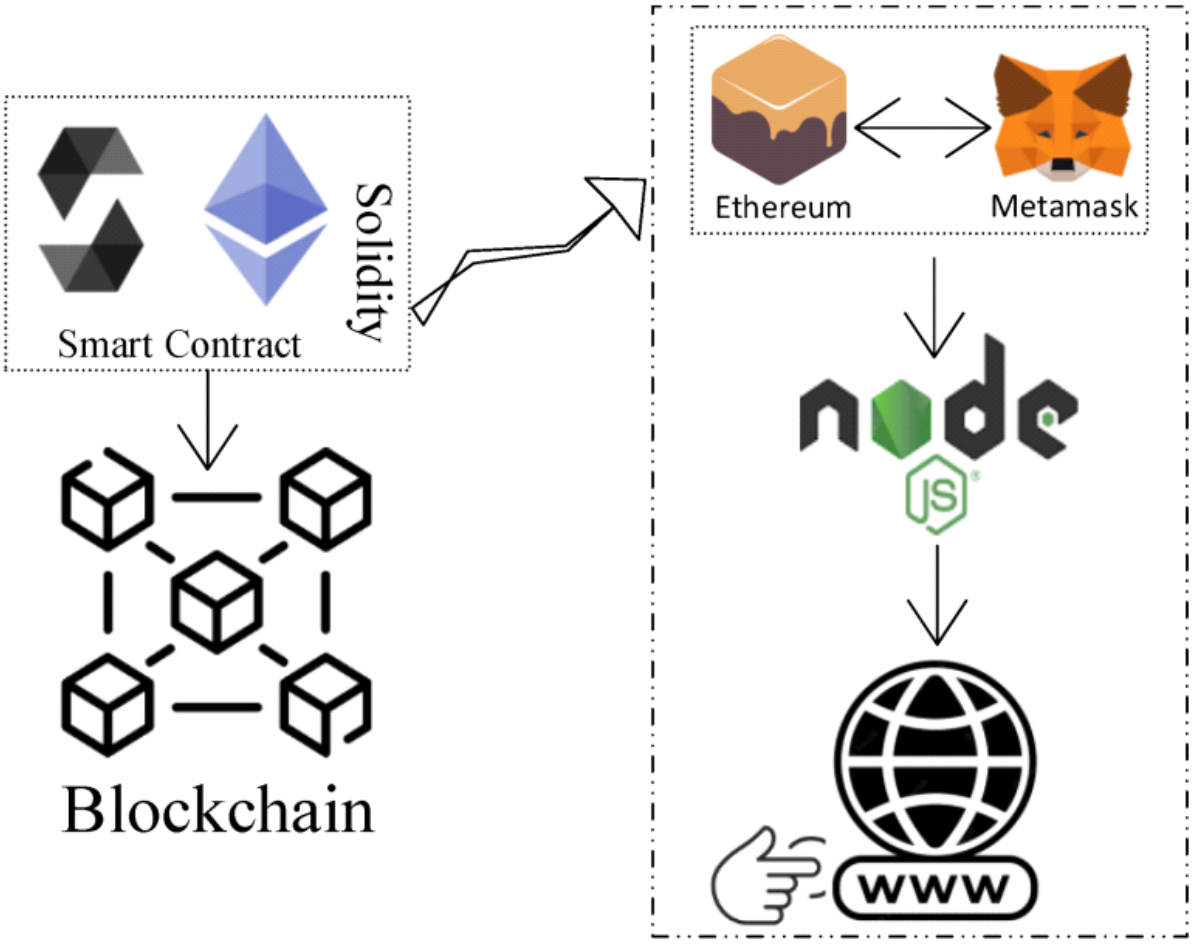
through the decentralized identity smart contract.As a service provider, I want to verify the authenticity of a user's decentralized identity to ensure the user is who they claim to be.

Story 3

As a developer, I want to easily integrate the Ethereum DID smart contract into my applications and services for secure user authentication and data access.

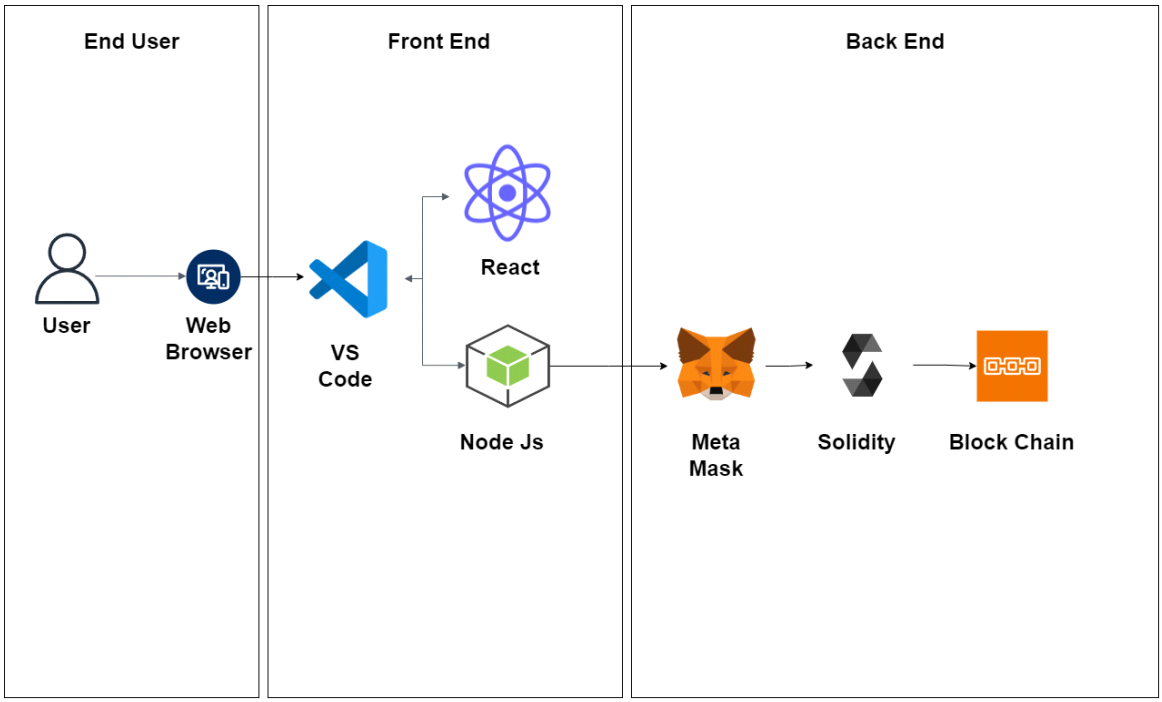
5.2 Solution Architecture





6.PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



GENERAL ARCHITECTURE

6.2 Sprint Planning and Estimation

The technical architecture of the system would involve designing and implementing the various components required for the biometric security system. This would include the hardware and software infrastructure needed to capture and authenticate biometric data, integrate with the blockchain platform, and handle the processing and storage of voting data.

6.2 Sprint Planning & Estimation:

Sprint planning involves breaking down the project into smaller, manageable tasks and determining the order in which they will be completed. Each task is estimated in terms of effort or time required for completion. The team can then plan the sprints, which are time-boxed iterations, to work on these tasks.

6.3 Sprint Delivery Schedule:

The sprint delivery schedule outlines the timeline for completing each sprint and delivering the planned functionality. It helps the team stay on track and ensures that progress is made incrementally throughout the project. The schedule may include milestones, such as the completion of specific features or the integration of different components.

Planning and scheduling are crucial for managing the project effectively and ensuring timely delivery. It allows for better coordination, resource allocation, and tracking of progress.

7.CODING AND SOLUTING

7.1 Feature1

Smart Contract(Solidity)

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

contract BallotBox {
    address public owner;

    struct Voter {
        bytes32 biometricData;  // Encrypted biometric data
        bool hasVoted;          // Indicates if the voter has cast a vote
    }

    struct Candidate {
        string name;
        uint256 voteCount;
    }

    string public electionName;
    uint256 public registrationDeadline;
    uint256 public votingDeadline;

    Candidate[] public candidates;

    mapping(address => Voter) public voters;

    event VoteCast(address indexed voter, uint256 candidateIndex);

   modifier onlyOwner() {
```

```
    require(msg.sender == owner, "Only the owner can call this function.");
    _;
}

modifier canVote() {
    require(block.timestamp < votingDeadline, "Voting has ended.");
    require(block.timestamp < registrationDeadline, "Registration has ended.");
    require(!voters[msg.sender].hasVoted, "You have already voted.");
    _;
}

constructor(
    string memory _electionName,
    uint256 _registrationDeadline,
    uint256 _votingDeadline,
    string[] memory _candidateNames
) {
    owner = msg.sender;
    electionName = _electionName;
    registrationDeadline = _registrationDeadline;
    votingDeadline = _votingDeadline;

    for (uint256 i = 0; i < _candidateNames.length; i++) {
        candidates.push(Candidate({
            name: _candidateNames[i],
            voteCount: 0
        }));
    }
}
```

```
    function registerVoter(bytes32 _encryptedBiometricData) public canVote {
        voters[msg.sender] = Voter({
            biometricData: _encryptedBiometricData,
            hasVoted: false
        });
    }

    function castVote(uint256 _candidateIndex) public canVote {
        require(_candidateIndex < candidates.length, "Invalid candidate index.");
        require(voters[msg.sender].biometricData != 0, "You must register first.");

        voters[msg.sender].hasVoted = true;

        candidates[_candidateIndex].voteCount++;

        emit VoteCast(msg.sender, _candidateIndex);
    }
}
```

7.2Feature 2

Frond end(Java Script)

```
import React, { useState } from "react";
import { Button, Container, Row, Col } from 'react-bootstrap';
import '../../node_modules/bootstrap/dist/css/bootstrap.min.css';
import { contract } from "./connector";

function Home() {
 const [Id, setId] = useState("");
 const [name, setName] = useState("");
```

```
const [email, setEmail] = useState("");
const [addr, setAddr] = useState("");
const [mAddr, setmAddr] = useState("");
const [data, setdata] = useState("");
  const [Wallet, setWallet] = useState("");



const handleId = (e) => {
 setId(e.target.value)
}

const handleName = (e) => {
 setName(e.target.value)
}

const handleEmail = (e) => {
 setEmail(e.target.value)
}

const handleAddr = (e) => {
 setAddr(e.target.value)
}

const handleRegisterIdentity = async () => {
 try {
  let tx = await contract.registerIdentity(Id.toString(), name,  email, addr)

 let wait =  await tx.wait()
 alert(wait.transactionHash)
```

```
 console.log(wait);
 } catch (error) {
  alert(error)
 }
}

const handleMetamaskAddr = (e) => {
 setmAddr(e.target.value)
}



const handleAddrDetails = async () => {
 try {
  let tx = await contract.getIdentityDetails(mAddr)
  let arr = []
  tx.map(e => arr.push(e))
  setdata(arr)
  // alert(tx)
  console.log(tx);
 } catch (error) {
  alert(error)
 }
}

  const handleWallet = async () => {
     if (!window.ethereum) {
        return alert('please install metamask');
     }

     const addr = await window.ethereum.request({
```

```
      method: 'eth_requestAccounts',
    });

    setWallet(addr[0])

  }

 return (
  <div>
   <h1 style={{ marginTop: "30px", marginBottom: "80px" }}>Identity On Blockchain</h1>
     {!Wallet ?

       <Button onClick={handleWallet} style={{ marginTop: "30px", marginBottom: "50px" }}>Connect Wallet </Button>
       :
       <p style={{ width: "250px", height: "50px", margin: "auto", marginBottom: "50px", border: '2px solid #2096f3' }}>{Wallet.slice(0, 6)}....{Wallet.slice(-6)}</p>
     }
   <Container>
    <Row>
     <Col style={{marginRight:"100px"}}>
      <div>

     <input style={{ marginTop: "10px", borderRadius: "5px" }} onChange={handleId} type="number" placeholder="Enter Identity ID" value={Id} /> <br />

     <input style={{ marginTop: "10px", borderRadius: "5px" }} onChange={handleName} type="string" placeholder="Enter Name" value={name} /> <br />

     <input style={{ marginTop: "10px", borderRadius: "5px" }} onChange={handleEmail} type="string" placeholder="Enter Email" value={email} /><br />
```

```
    <input style={{ marginTop: "10px", borderRadius: "5px" }} onChange={handleAddr} type="string" placeholder="Enter Address" value={addr} /><br />


    <Button onClick={handleRegisterIdentity} style={{ marginTop: "10px" }} variant="primary">Register Identity</Button>

    </div>

   </Col>

   <Col>

    <div>

    <input style={{ marginTop: "10px", borderRadius: "5px" }} onChange={handleMetamaskAddr} type="string" placeholder="User Metamask address" value={mAddr} /><br />


    <Button onClick={handleAddrDetails} style={{ marginTop: "10px" }} variant="primary">Get Identity Details</Button>

    {data ? data?.map(e => {

      return <p>{e.toString()}</p>

    }) : <p></p>}

    </div>

   </Col>

  </Row>


 </Container>


</div>

)

}


export default Home;
```

Contract ABI (Application Binary Interface):

The abi variable holds the ABI of an Ethereum smart contract. ABIs are essential for encoding and decoding function calls and data when interacting with the Ethereum blockchain.

MetaMask Check:

The code first checks whether the MetaMask wallet extension is installed in the user's browser. If MetaMask is not detected, it displays an alert notifying the user that MetaMask 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 Web3Provider, which connects to the user's MetaMask wallet and provides access to Ethereum. It creates a signer to interact with the Ethereum blockchain 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 Ethereum smart contract through MetaMask and ethers.js. It configures the necessary Ethereum provider and signer for communication with the blockchain and sets up a contract object for executing functions and fetching data from the specified contract address using the provided ABI.

8. PERFORMANCE TESTING

8.1 Performance Matrix

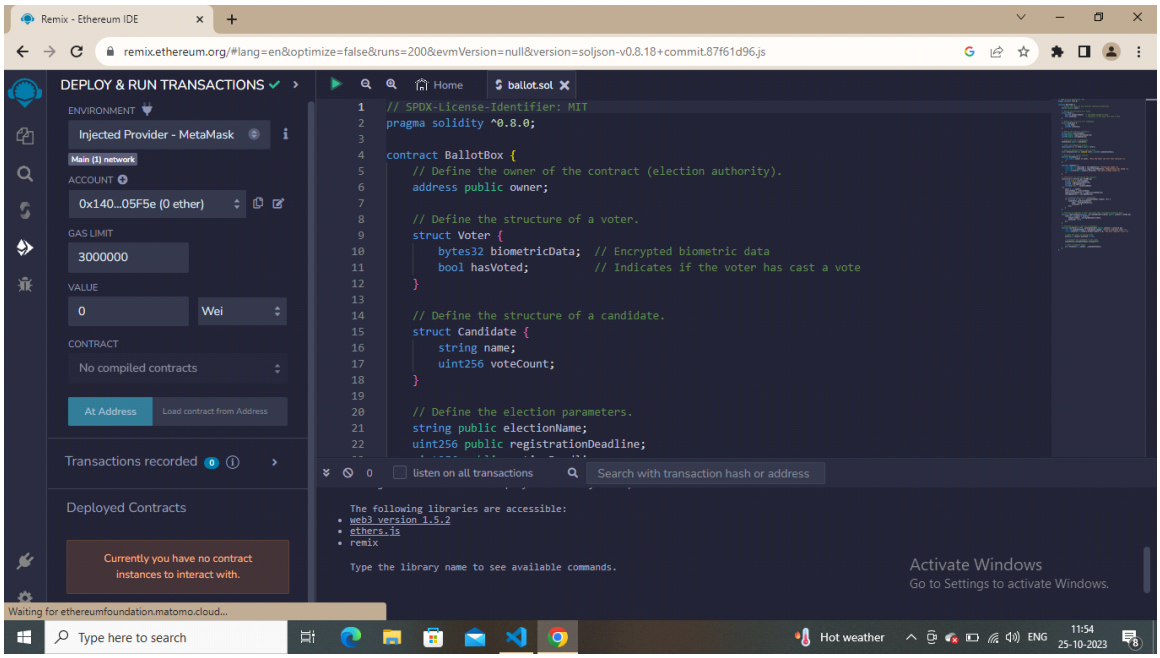
Performance metrics are essential for evaluating the performance of a system. In addition to the metrics I mentioned earlier, there are a few more worth considering:

- Error Handling: This metric focuses on how well the system handles errors and exceptions. It helps assess the system's robustness and ability to recover from failures.
- Scalability: Scalability measures how well the system can handle an increasing workload or user base. It helps determine if the system can handle future growth without compromising performance.
- Load Testing: This metric involves testing the system's performance under high loads to evaluate its stability and responsiveness.
- Stress Testing: Stress testing pushes the system beyond its normal operating limits to determine its breaking point. It helps identify any weaknesses or bottlenecks in the system.
- Endurance Testing: Endurance testing assesses the system's performance over an extended period to ensure it can handle sustained usage without degradation.
- Response Time: This measures the time it takes for the system to respond to a request. It helps assess the system's speed and responsiveness.
- Throughput: This metric measures the number of transactions or requests the system can handle within a given time frame. It indicates the system's capacity and scalability.
- Concurrent Users: This metric measures the number of users that can simultaneously access and use the system without degrading its performance. It helps determine the system's ability to handle concurrent loads.
- Error Rate: This metric measures the percentage of failed or erroneous transactions. It helps assess the system's reliability and stability.

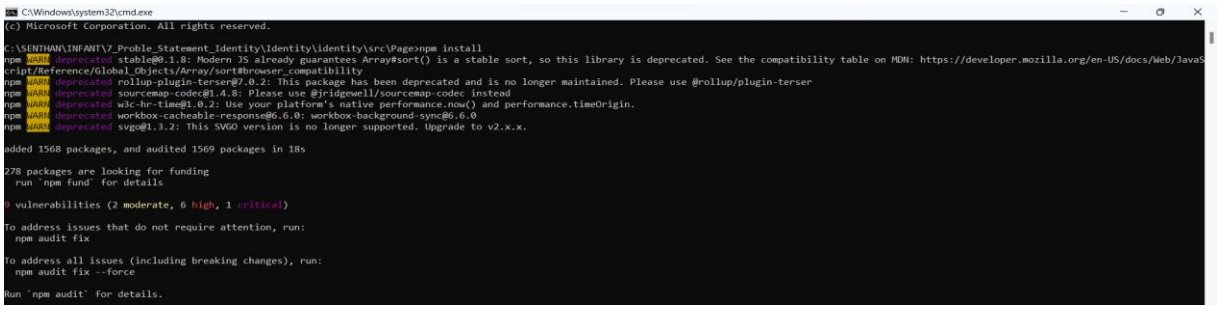
- Resource Utilization: This metric measures the usage of system resources such as CPU, mmory, and network bandwidth. It helps identify any resource bottlenecks or inefficiencies.

9.RESULTS

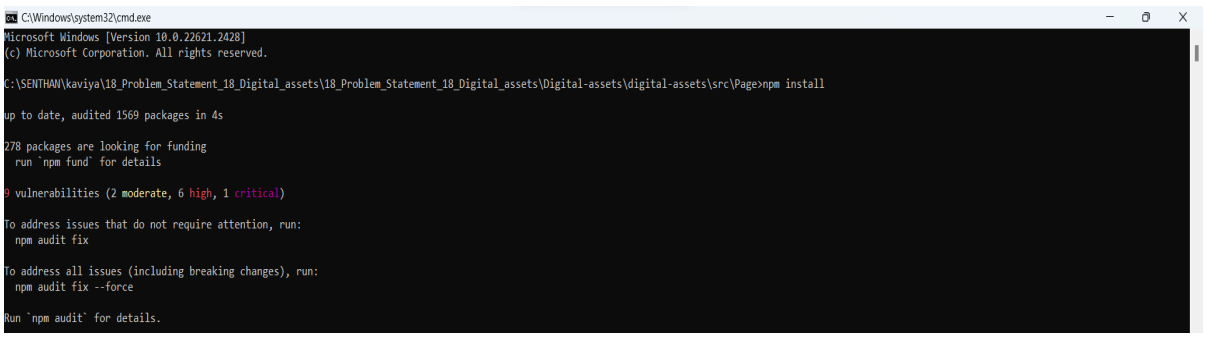
9.1 OUTPUT SCREENSHOTS



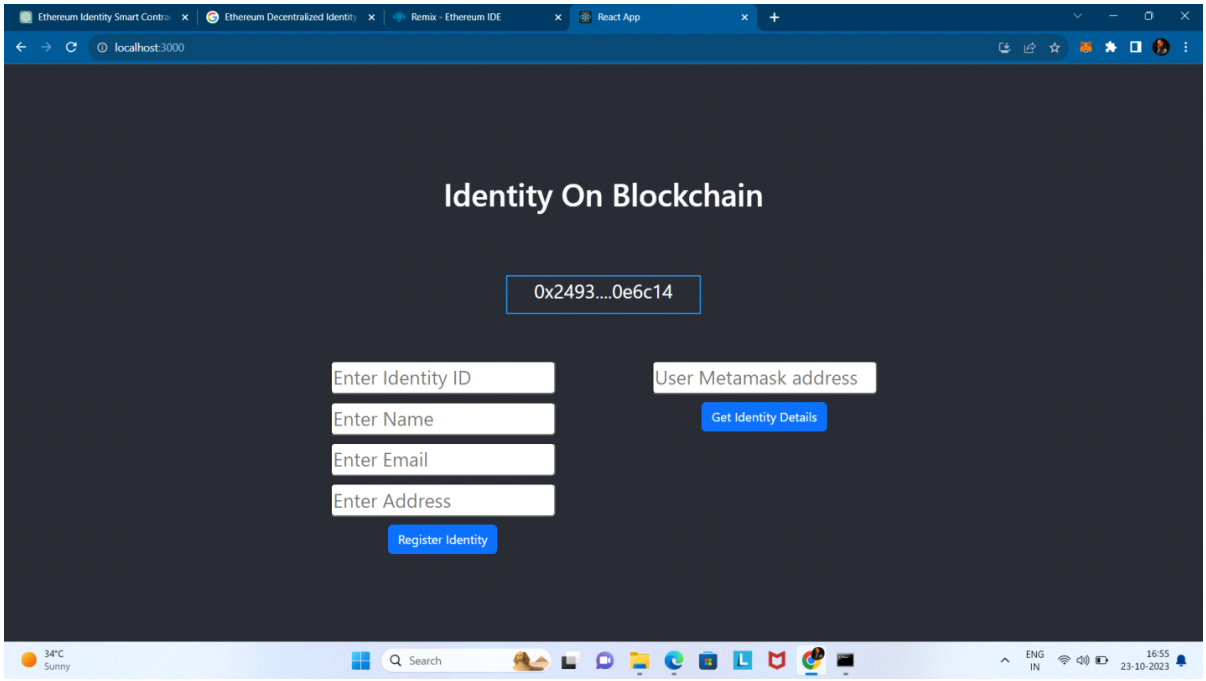
CREATING A SMART CONTACT



INSTALLING DEPENDENCIES 1



INSTALLING DEPENDENCIES 2



WEB PAGE OUTPUT

10.ADVANTAGES AND DISADVANTAGES

10.1 ADVANTAGES

- Increased security - Biometrics like fingerprint scanning or facial recognition can help verify voter identity and prevent voter fraud. This strengthens the integrity of the voting process.
- Convenience - Once a biometric identity is established, voters may find it faster and easier to vote without having to present IDs or remember passwords/PINs. This can increase voter turnout.
- Accuracy - Biometrics are unique to each person so they eliminate issues like duplicate registrations or fake IDs. This improves the accuracy of voter rolls.
- Accessibility - Biometrics can make it easier for some disabled voters who may have trouble with other forms of ID to independently cast their votes..

10.2 DISADVANTAGES

- Expense - Implementing biometric systems requires new hardware, software, and worker training which can be costly, especially for small jurisdictions. Ongoing maintenance is also required.
- Exclusion - Biometrics may exclude certain groups of people, like the elderly who may not enroll their fingerprints properly or those with disabilities that affect their biometric data. Alternate accommodations would be needed.
- Privacy concerns - Some fear biometric data collection is an invasion of privacy and increases government surveillance. The data would need proper cybersecurity protections.

- Technical difficulties - Sensors could malfunction or biometrics may not properly scan for some people. Having backup options like IDs available would be important to avoid disenfranchising voters.
- Centralized databases - Storing everyone's biometric profiles in centralized databases creates security/privacy risks and a single point of failure. Distributed ledger technology like blockchain may mitigate this.

11. CONCLUSION

In conclusion, implementing a biometric security system for a voting platform has the potential to enhance security and reduce the risk of fraudulent voting. Biometric authentication methods like fingerprint, facial recognition, or iris scanning can provide a higher level of identity verification compared to traditional methods. However, it's essential to address privacy concerns, ensure accessibility for all voters, and maintain a balance between security and ease of use. Regular testing, auditing, and oversight are crucial to ensure the integrity of the system. Ultimately, the success of a biometric security system in a voting platform depends on a comprehensive and well-executed approach that takes into account various technical, legal, and ethical considerations.

12. FUTURE SCOPE

1. Enhanced Security: Biometric authentication can significantly enhance the security of voting systems by ensuring that only eligible voters can cast their ballots. This can help prevent identity theft and voter fraud.

2. Voter Convenience: Biometrics can make the voting process more convenient for voters as they won't need to carry physical IDs or remember passwords. This can potentially increase voter turnout.

3. Accessibility: Biometric systems can be designed to accommodate individuals with disabilities, making it more inclusive and accessible for all citizens.

4. Accuracy: Biometric verification can help reduce errors in the voting process and ensure that each vote is counted accurately.

5. Remote Voting: Biometric authentication can enable secure remote voting, which can be especially relevant during situations like pandemics or for citizens living abroad.

6. Transparency: To gain public trust, it's crucial to implement biometric systems that are transparent and auditable, allowing voters to verify the integrity of their votes.

7. Privacy Concerns: Addressing privacy concerns is important. Biometric data must be securely stored and protected to prevent misuse.

8. Regulatory Framework: The future scope depends on the development of clear and comprehensive regulations to govern the use of biometrics in voting to ensure fairness and security.

13.APPENDIX

SOURCE CODE

/ SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

```
contract BallotBox {
    address public owner;

    struct Voter {
        bytes32 biometricData;  // Encrypted biometric data
        bool hasVoted;          // Indicates if the voter has cast a vote
    }

    struct Candidate {
        string name;
        uint256 voteCount;
    }

    string public electionName;
    uint256 public registrationDeadline;
    uint256 public votingDeadline;

    Candidate[] public candidates;

    mapping(address => Voter) public voters;

    event VoteCast(address indexed voter, uint256 candidateIndex);

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

    modifier canVote() {
        require(block.timestamp < votingDeadline, "Voting has ended.");
```

```
    require(block.timestamp < registrationDeadline, "Registration has ended.");
    require(!voters[msg.sender].hasVoted, "You have already voted.");
    _;
}

constructor(
    string memory _electionName,
    uint256 _registrationDeadline,
    uint256 _votingDeadline,
    string[] memory _candidateNames
) {
    owner = msg.sender;
    electionName = _electionName;
    registrationDeadline = _registrationDeadline;
    votingDeadline = _votingDeadline;

    for (uint256 i = 0; i < _candidateNames.length; i++) {
        candidates.push(Candidate({
            name: _candidateNames[i],
            voteCount: 0
        }));
    }
}

function registerVoter(bytes32 _encryptedBiometricData) public canVote {
    voters[msg.sender] = Voter({
        biometricData: _encryptedBiometricData,
        hasVoted: false
    });
}
```

```
  function castVote(uint256 _candidateIndex) public canVote {
      require(_candidateIndex < candidates.length, "Invalid candidate index.");
      require(voters[msg.sender].biometricData != 0, "You must register first.");

      voters[msg.sender].hasVoted = true;

      candidates[_candidateIndex].voteCount++;

      emit VoteCast(msg.sender, _candidateIndex);
   }
}
```

Digitalassest.sol