# ROHINI College of Engineering and Technology

## Naan Mudhalvan Project

## Biometric Security System for Voting Platform

| | |
|---|---|
| **NM Team ID** | NM2023TMID07060 |
| **Project Name** | Biometric Security System for Voting Platform |

| | Name | NM ID |
|---|---|---|
| **Team Leader** | Harish S | 168874C6C3E13ACA77B4BE7E2D09B108 |
| **Team Members** | Anton Nekesh A | BAF1FF0B4808E6A97D94E9353E9CBBDA |
| | Kabilesh C M | C807C76B3E8503FA3DF4004CC32EEC90 |
| | Aravind S | F3A5CF946693212AF9B464ADE0811217 |

# Index

## INTRODUCTION

Introducing a biometric security system for a voting platform is like giving your democracy a high-tech makeover! Imagine a future where your vote is not just a mark on paper but a unique, irrefutable imprint of your identity. Biometric security brings in a level of accuracy and authentication that traditional methods may lack.

### Project Overview

Combining blockchain technology with a biometric security system for a voting platform is like adding an extra layer of trust and transparency to the democratic process. The primary aim of this project is to enhance the security and integrity of the voting process by implementing a biometric security system in a voting platform. By utilizing biometric authentication methods, the project seeks to ensure a more accurate and secure identification of voters, mitigating the risks associated with identity fraud and unauthorized voting.

### Purpose

The purpose of a Biometric Security System for a Voting Platform is to fortify the democratic process by introducing a robust and secure means of verifying voter identity. By leveraging unique biometric markers such as fingerprints or facial features, the system aims to eliminate the risks associated with identity fraud and multiple voting instances. This advanced security layer not only enhances the accuracy of voter authentication but also contributes to building trust and confidence in the integrity of electoral outcomes. The ultimate goal is to streamline the voting process, prevent fraudulent activities, and ensure that each citizen's voice is accurately and securely represented in the democratic arena.

## LITERATURE SURVEY

The literature survey of Biometric Security Systems for Voting Platforms delves into existing research and studies to assess the current landscape of biometric applications in electoral processes. Scholars explore the effectiveness of biometric authentication methods, such as fingerprints, facial recognition, and iris scans, in mitigating identity fraud and enhancing the security of voting platforms. The survey delves into the integration of biometric systems with technologies like blockchain to

ensure tamper-resistant record-keeping and transparent electoral processes. Researchers also examine the challenges and opportunities presented by the adoption of biometric security, addressing issues of privacy, accessibility, and technological feasibility. The synthesis of existing literature informs the development of a comprehensive and informed approach to implementing biometric security measures in voting platforms.

**Existing Problem**

The existing problems in Biometric Security Systems for Voting Platforms include concerns about the potential compromise of voter privacy and the security of biometric data. Issues related to the accuracy and reliability of biometric authentication methods, especially in diverse demographic settings, pose challenges. Additionally, the integration of biometric systems with existing voting infrastructure may face resistance due to cost implications and technological constraints. Ensuring accessibility for all voters and addressing regulatory and ethical considerations surrounding biometric data usage remain critical hurdles. A comprehensive understanding of these challenges is crucial for developing effective solutions that balance security, privacy, and inclusivity in the context of voting platforms.

**References**


- ✓ P. Wolf, R. Nackerdien and D. Tuccinardi, "Introducing Electronic Voting: Essential Considerations", Int. Inst. Democr. Elect. Assist., no. December, pp. 39, 2011.
- ✓ H. Timeline, D. Recording, E. Voting, T. Australian, A. S. Ballot and H. Hollerith, "Historical Timeline Electronic Voting Machines and Related Voting Technology", Chief Eng., pp. 1-7, 2011.
- ✓ B. Simons and D. W. Jones, "Internet voting in the U.S.", Commun. ACM, vol. 55, no. 10, pp. 68-77, 2012.
- ✓ P. A. Colinvaux, "The past and future of Internet Voting", Sci. Am., no. 260, pp. 101-108, 1989.
- ✓ M. Górny, "I-voting - opportunities and threats. Conditions for the effective implementation of Internet voting on the example of Switzerland and Estonia", Przeglad Politol., no. 1, pp. 133-146, 2021.
- ✓ "Difference Between Client-Server and Peer-to-Peer Network", Website, 2017, [online] Available: https://techdifferences.com!difference-between-client-server-and-peer-to-peer-network.html.

- ✓ R. Shivers, M. A. Rahman, M. J. H. Faruk, H. Shahriar, A. Cuzzocrea and V. Clincy, "Ride-Hailing for Autonomous Vehicles: Hyperledger Fabric-Based Secure and Decentralize Blockchain Platform", Proc. − 2021 IEEE Int. Conf. Big Data Big Data 2021, pp. 5450-5459, 2021.
- ✓ M. J. Hossain Faruk, H. Shahriar, M. Valero, S. Sneha, S. Ahamed and M. Rahman, "Towards Blockchain-Based Secure Data Management for Remote Patient Monitoring", IEEE nternational Cnference Degital Heal., pp. 299-308, 2021.
- ✓ M. J. Hossain Faruk, S. Hossain and M. Valero, "EHR Data Management: Hyperledger Fabric-based Health Data Storing and Sharing", Fall 2021 Symp. Student Sch., 2021.
- ✓ M. J. Hossain Faruk, S. Hossain and M. Valero, "Students Certification Management (SCM): Hyperledger Fabric-Based Digital Repository", CCSE Comput. Showc. Day, 2021.
- ✓ R. Bulut, A. Kantarci, S. Keskin and S. Bahtiyar, "Blockchain-Based Electronic Voting System for Elections in Turkey", UBMK 2019 - Proceedings 4th Int. Conf. Comput. Sci. Eng., pp. 183-188, 2019.
- ✓ D. R. Joseph, "Hyperledger Architecture Volume II - Smart Contracts", Gene, vol. 17, no. 3, pp. 341-344, 1982.
- ✓ K. Yamashita, Y. Nomura, E. Zhou, B. Pi and S. Jun, "Potential Risks of Hyperledger Fabric Smart Contracts", IWBOSE 2019 − 2019 IEEE 2nd Int. Work. Blockchain Oriented Softw. Eng., pp. 1-10, 2019.
- ✓ S. Al-Maaitah, M. Qatawneh and A. Quzmar, "E-Voting System Based on Blockchain Technology: A Survey", 2021 Int. Conf. Inf. Technol. Icit 2021 - Proc., pp. 200-205, 2021.
- ✓ Z. Zhao, "Comparison of Hyperledger Fabric and Ethereum Blockchain", 2022 IEEE Asia-Pacific Conf. Image Process. Electron. Comput. IPEC 2022, pp. 584-587, 2022.
- ✓ "Hyperledger Architecture volume 1: Introduction to hyperledger business blockchain consensus models", Hyperledger Archit., vol. 1, pp. 1-7, 2016.
- ✓ E. Androulaki et al., "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains", Proc. 13th EuroSys Conf. EuroSys 2018, vol. 2018-Janua, 2018.
- ✓ "What is Hyperledger Fabric?", Ibm, 2021, [online] Available: https://www.ibm.com/topics/hyperledger.
- ✓ M. Dastbaz, E. Halpin and S. Wright, "Emerging technologies and the human rights challenge of rapidly expanding state surveillance capacities", Strateg. Intell. Manag. Natl. Secur. Imp. Inf. Commun. Technol., 2013.
- ✓ S. Sumner, "Biometrics and The Future", You Sale, pp. 183-198, 2016.

## Problem Statement Definition

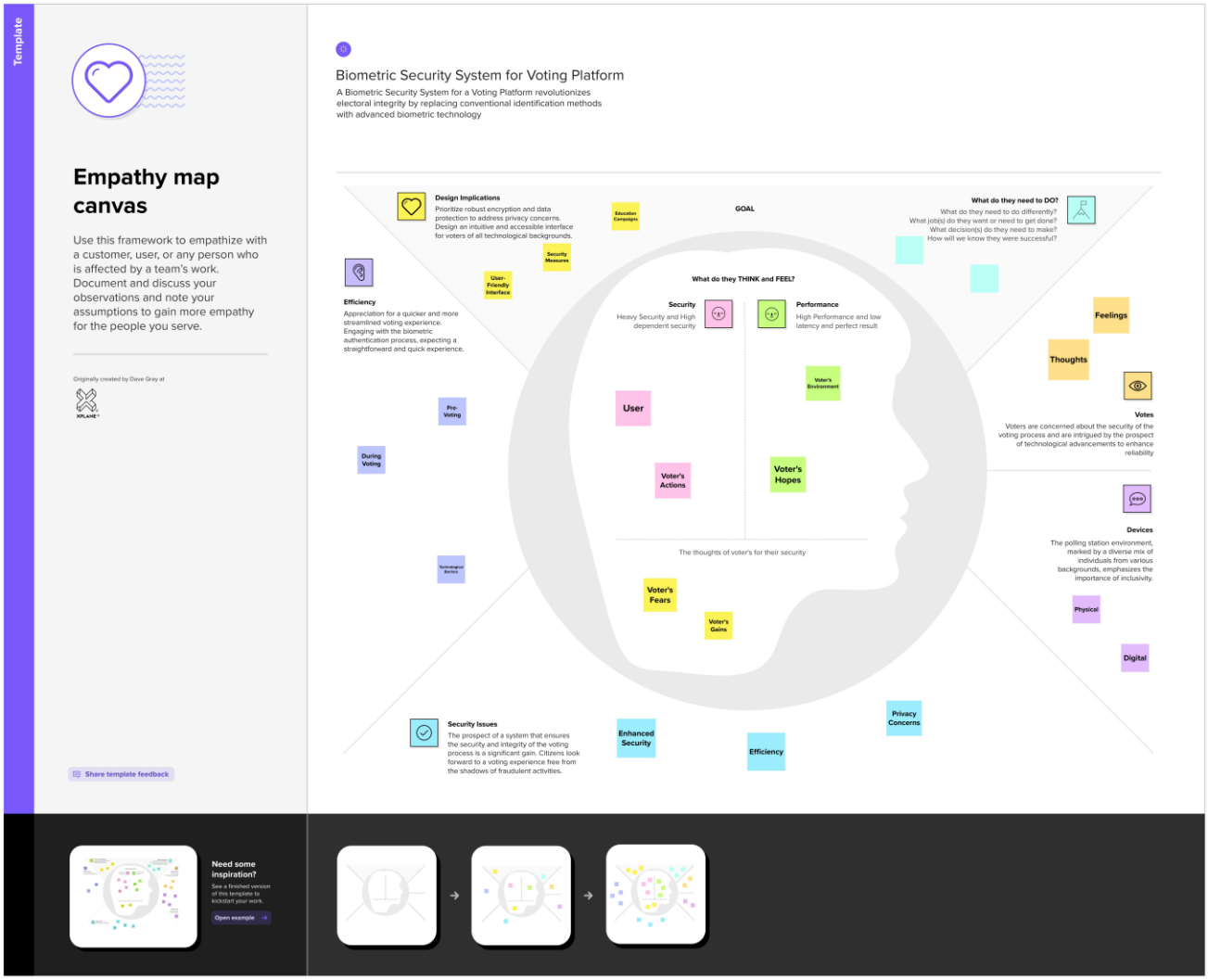
A Biometric Security System for a Voting Platform revolutionizes electoral integrity by replacing conventional identification methods with advanced biometric technology. Through the secure capture and verification of unique biological markers like fingerprints or facial features, this system ensures airtight authentication, virtually eliminating the risks of identity fraud and unauthorized access. By marrying cutting-edge security measures with user-friendly interfaces, it not only fortifies the democratic process but also instills confidence in voters, paving the way for a more trustworthy and inclusive electoral experience.

## IDEATION & PROPOSED SOLUTION

The key challenge lies in redefining the voting platform's security paradigm to address identity-related vulnerabilities. Leveraging biometric technology presents a compelling solution, ensuring a personalized and foolproof authentication process. Ideas include the integration of fingerprint scanners or facial recognition devices at polling stations, providing a seamless and secure means of verifying voters' identities. The proposed solution encompasses real-time verification algorithms, blockchain integration for tamper-proof data records, and encrypted databases to fortify the storage of biometric information. A user-friendly interface ensures accessibility, while redundant systems and failover mechanisms guarantee continuous operation. By prioritizing compliance with regulations and launching a public awareness campaign, our solution not only addresses current vulnerabilities but sets a new standard for trustworthy and inclusive electoral processes.

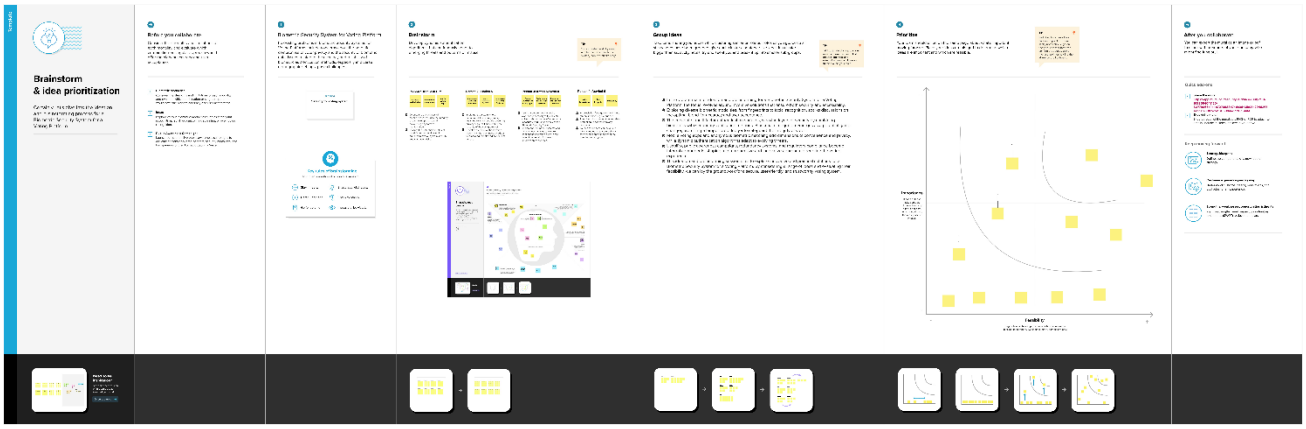
## Empathy Map Canvas

This comprehensive Empathy Map Canvas provides a holistic understanding of the user experience, their thoughts, emotions, actions, and expectations in the context of a Biometric Security System for a Voting Platform. It serves as a foundation for designing a system that not only meets functional requirements but also resonates with the users' needs and concerns, fostering a trustworthy and inclusive democratic process. Quotes echo hopes for secure, accurate, and fair elections, while influences from community discussions and media narratives shape overall perceptions. This empathetic exploration forms the basis for designing a system that not only addresses functional requirements but also aligns seamlessly with the diverse needs and expectations of voters, fostering trust and inclusivity in the democratic process.




## Ideation & Brainstorming

The ideation and brainstorming session aims to explore innovative and practical solutions for a Biometric Security System for a Voting Platform. By considering a range of ideas and evaluating their feasibility, we can lay the groundwork for a secure, user-friendly, and trustworthy voting system.



## REQUIREMENT ANALYSIS

The requirement analysis forms the foundation for developing a Biometric Security System that not only meets the functional needs of the voting platform but also addresses security, privacy, usability, and regulatory considerations essential for a trustworthy and reliable electoral process.

### Functional requirement

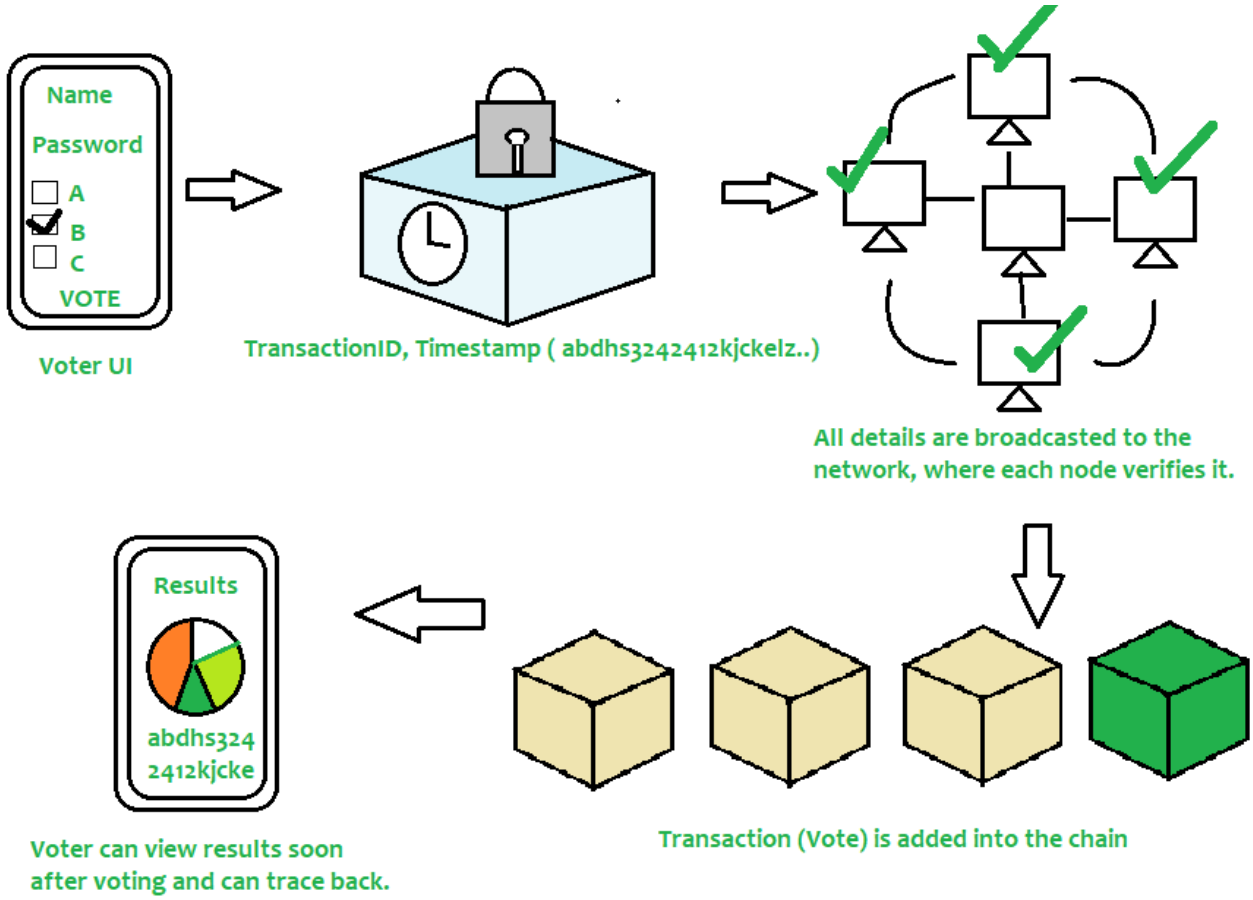
| Outline | Description |
|---|---|
| Voter Registration | The system should allow for the secure registration of voters, capturing and storing their biometric data, along with relevant personal information. |
| Biometric Data Collection | Enable the collection of various biometric data, such as fingerprints, facial features, or iris scans, during the voter registration process. |
| Real-time Biometric Authentication | Implement a real-time biometric authentication mechanism that compares the captured biometric data during voting with the stored records to verify the voter's identity. |
| Multi-Modal Biometrics Support | Support multiple biometric modalities to accommodate the preferences and physical conditions of a diverse voter population. |
| Multi-Factor Authentication | Integrate multi-factor authentication by combining biometric data with a secondary form of verification, such as a voter ID card or a one-time passcode. |
| User-Friendly Interface | Design an intuitive and user-friendly interface for both voters and election officials, ensuring a straightforward and efficient interaction with the system. |
| Blockchain Integration | Integrate blockchain technology to create an immutable and transparent ledger of votes, enhancing the security and verifiability of the voting process. |
| Data Encryption and Security | Implement robust encryption protocols to secure the biometric data during storage, transmission, and processing, preventing unauthorized access or tampering. |
| Auditing and Logging | Implement comprehensive auditing and logging mechanisms to track and record all system activities, ensuring transparency and accountability. |

**Non-Functional requirements**

| Outline | Description |
|---|---|
| Performance | The system must be capable of handling a large volume of simultaneous biometric authentication requests during peak voting times without significant degradation in performance. |
| Scalability | The system should be designed to scale horizontally to accommodate an increasing number of voters, ensuring seamless operation during elections of varying sizes. |
| Reliability | The system should have a high level of reliability, with minimal downtime and robust failover mechanisms to ensure continuous operation even in the face of unexpected issues. |
| Availability | Ensure high availability of the Biometric Security System, guaranteeing that voters can authenticate and cast their votes without disruptions throughout the entire voting period |
| Response Time | Define acceptable response times for biometric authentication to ensure a quick and efficient voting experience, minimizing wait times for voters. |
| Security | Implement stringent security measures to protect against unauthorized access, data breaches, and tampering of biometric information, ensuring the confidentiality and integrity of voter data. |
| Usability | Prioritize usability by designing an intuitive interface that is easy to navigate for voters and election officials, promoting a positive and accessible user experience. |
| Interoperability | Ensure interoperability with other election-related systems and technologies to facilitate data exchange and collaboration seamlessly. |
| Maintainability | Design the system with ease of maintenance in mind, enabling timely updates, patches, and modifications without significant disruptions to the voting process. |
| Ethical Considerations | Incorporate ethical considerations into the system design, addressing potential biases in biometric data and ensuring fairness and equal treatment of all voters. |
| Training and Support | Offer training programs and support for election officials to ensure the proper use and maintenance of the Biometric Security System. |

## PROJECT DESIGN

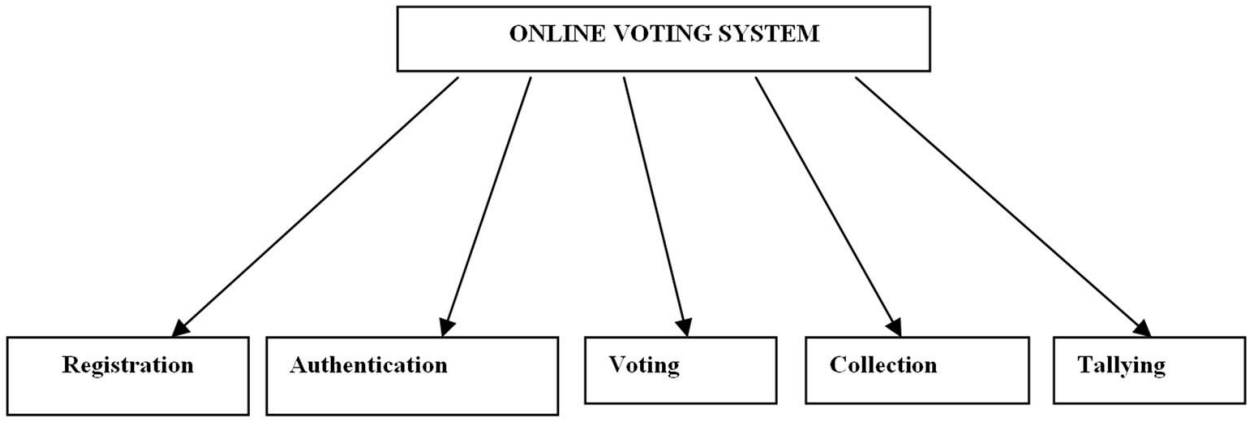
The integration of blockchain technology establishes an immutable ledger, safeguarding the integrity of every vote. Security is paramount, with robust encryption protocols, redundancy systems, and compliance with stringent privacy measures. Our user-centric approach emphasizes a seamless interface, accessibility features, and an extensive public awareness campaign to foster trust. Continuous improvement frameworks and adaptable technologies ensure the system evolves alongside emerging needs, reinforcing the democratic foundation of the voting process.




## Data Flow Diagrams & User Stories

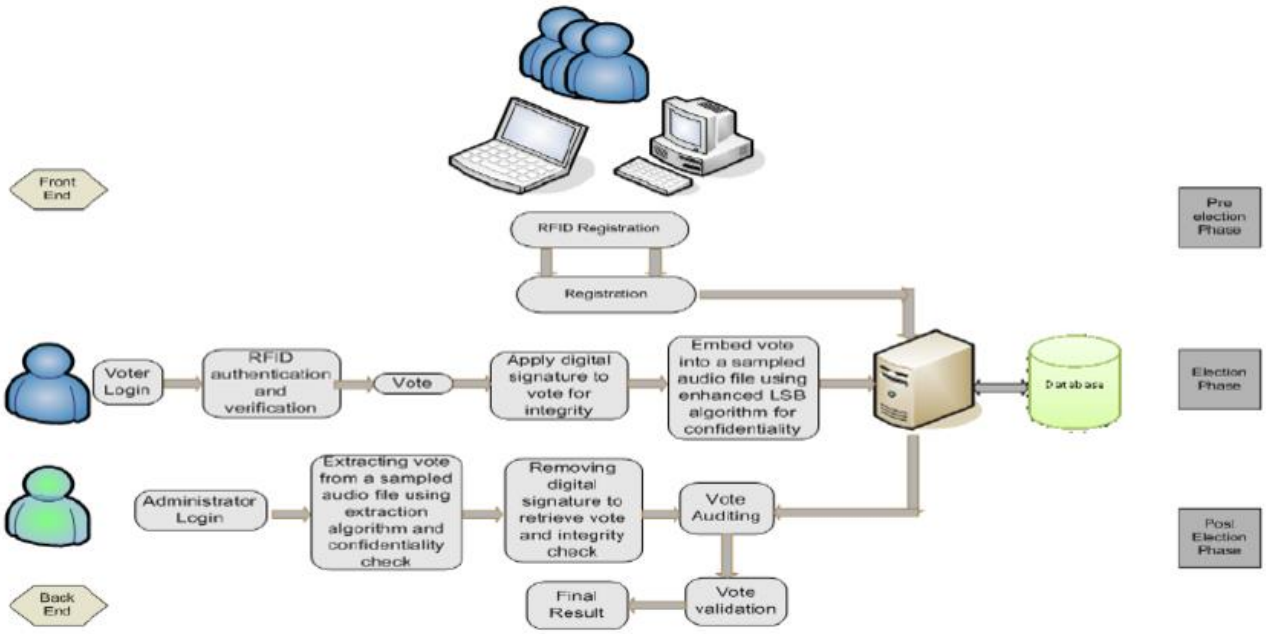
In the Data Flow Diagrams (DFD) of the Biometric Security System for a Voting Platform, the flow of information is meticulously depicted. The voter registration initiates the process, with biometric data seamlessly flowing into the system. Real-time authentication processes occur, leveraging multi-modal biometrics and multi-factor verification. The integration of blockchain forms a secure and transparent layer, recording votes as they traverse the system. Data encryption

protocols safeguard the information at every stage, and a user-friendly interface ensures a smooth interaction for both voters and election officials. The DFD encapsulates the intricate pathways of data, emphasizing security, transparency, and efficiency in the electoral process.




## Solution Architecture

The solution architecture for the Biometric Security System envisions a robust framework that seamlessly integrates cutting-edge technologies. A modular structure supports key components, including voter registration, biometric data collection, real-time authentication, and blockchain integration. Multi-modal biometrics and multi-factor authentication enhance security, while data encryption protocols fortify the confidentiality of sensitive information



## PROJECT PLANNING & SCHEDULING

The project planning and scheduling framework ensures a systematic and efficient development process for the Biometric Security System, balancing flexibility, stakeholder engagement, and the delivery of a secure and reliable voting platform.

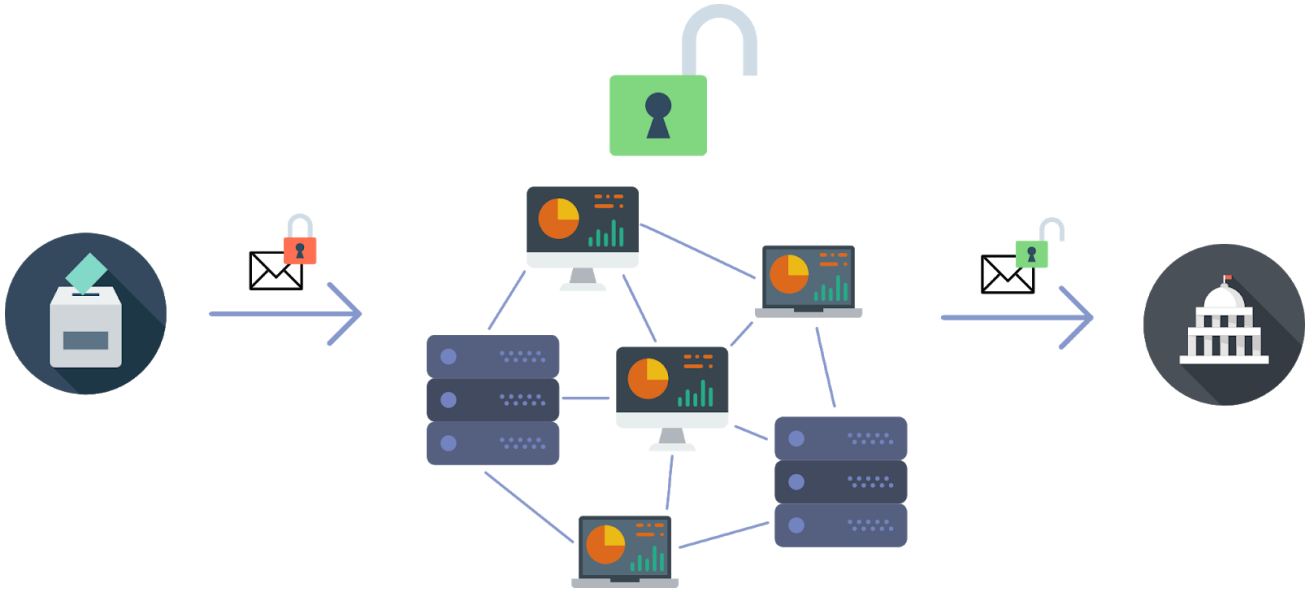
The scope has been precisely defined, stakeholders engaged, and resources allocated efficiently. A detailed work breakdown structure (WBS) maps out the tasks, their dependencies, and critical milestones. Time estimation, Gantt chart creation, and risk analysis form a robust foundation, ensuring a realistic and achievable project timeline.

### Technical Architecture

The Technical Architecture of the Biometric Security System for a Voting Platform is intricately designed for efficiency, security, and adaptability. It incorporates modules for voter registration, real-time biometric authentication, and blockchain integration, supporting multi-modal biometrics and multi-factor authentication.

Robust encryption and security measures safeguard sensitive data, while user-friendly interfaces prioritize accessibility. Redundancy systems ensure continuous operation, and compliance and privacy modules address regulatory requirements.

The technical architecture not only leverages cutting-edge technologies but also prioritizes scalability, testing, and continuous improvement, shaping a resilient foundation for a trustworthy and advanced voting platform



## Sprint Planning & Estimation

Sprint planning and estimation are iterative processes that involve continuous communication, collaboration, and adaptation to ensure the successful development of a Biometric Security System for a Voting Platform.

| | | |
|---|---|---|
| **1.** Backlog Refinement | **5.** Capacity Planning | **9.** Burndown Charts |
| **2.** Sprint Planning Meeting | **6.** Velocity Calculation | **10.** Retrospective Meeting |
| **3.** Task Breakdown | **7.** Prioritization | **11.** Adaptation and Iteration |
| **4.** Estimation | **8.** Daily Standups | **12.** Review and Demo |

## Sprint Delivery Schedule

- **Foundation and Planning (2 weeks)**
  - ✓ Define project scope and objectives
  - ✓ Set up the development environment
  - ✓ Establish communication channels
- **Voter Registration Module (3 weeks)**
  - ✓ Develop voter registration module
  - ✓ Implement biometric data collection mechanisms
  - ✓ Integrate data encryption and security measures
- **Real-time Authentication Module (3 weeks)**
  - ✓ Develop real-time authentication module
  - ✓ Implement multi-modal biometrics support
- **Blockchain Integration (4 weeks)**
  - ✓ Integrate blockchain technology for transparent vote ledger
  - ✓ Begin documentation of blockchain features and security protocols
- **Usability and Interface Enhancement (2 weeks)**
  - ✓ Design and implement user-friendly interfaces
  - ✓ Conduct usability testing
- **Integration and Redundancy Systems (3 weeks)**
  - ✓ Ensure seamless integration with existing systems
  - ✓ Implement redundancy and failover mechanisms
- **Compliance and Privacy Measures (2 weeks)**
  - ✓ Implement compliance modules for regulatory requirements
  - ✓ Enhance privacy measures for biometric data
- **Testing and Quality Assurance (3 weeks)**
  - ✓ Conduct rigorous testing of the entire Biometric Security System
  - ✓ Implement quality assurance measures

- **Deployment Preparation (2 weeks)**
  - ✓ Plan for the deployment of the Biometric Security System
  - ✓ Develop deployment strategies and documentation
- **Final Testing and Optimization (2 weeks)**
  - ✓ Conduct final testing of the entire system
  - ✓ Optimize performance and address any remaining issues

## CODING & SOLUTIONING

In the coding and solutioning phase of the Biometric Security System for a Voting Platform, we embark on the intricate process of transforming conceptual designs into functional reality. The development team begins by coding the various modules, adhering to best coding practices, and utilizing the chosen programming languages and frameworks.

### Home Page Code

```
import React, { useState } from "react";
import { Button, Container, Row, Col } from 'react-bootstrap';
import 'bootstrap/dist/css/bootstrap.css';
import { contract } from "./connector";
function Home() {
  const [Wallet, setWallet] = useState("");
  const [CandidateIndex, setCandidateIndex] = useState("");
  const [VoterData, setVoterData] = useState("");
  const [CandidateIndexed, setCandidateIndexed] = useState("");
  const [CandidatesData, setCandidatesData] = useState("");
  const [RegDeadline, setRegDeadlne] = useState("");
  const [VoteDeadlne, setVoteDeadlne] = useState("");
  const [Election, setElection] = useState("");
  const handleCandidateIndex = (e) => {
    setCandidateIndex(e.target.value)
```

```
}
  const handleCastVote = async () => {
    try {
      let tx = await contract.castVote(CandidateIndex.toString())
      let wait = await tx.wait()
      console.log(wait);
      alert(wait.transactionHash)
    } catch (error) {
      alert(error)
    }
  }
  const handleVoterBiometricData = (e) => {
    setVoterData(e.target.value)
  }
 const handleRegisterVoter = async () => {
    try {
      let tx = await contract.registerVoter(VoterData)
      let wait = await tx.wait()
      console.log(wait)
      alert(wait.transactionHash)
    } catch (error) {
      alert(error)
    }
  }
  const handleCandidateIndexs = (e) => {
    setCandidateIndexed(e.target.value)
  }
```

```
const handleCandidate = async () => {
  try {
    let tx = await contract.candidates(CandidateIndexed.toString())
    setCandidatesData(tx)
    console.log(tx);
    // alert(wait.transactionHash)
  } catch (error) {
    alert(error)
  }
}
const handleRegdeadline = async () => {
  try {
    let tx = await contract.registrationDeadline()
    setRegDeadlne(tx)
    console.log(tx);
    // alert(wait.transactionHash)
  } catch (error) {
    alert(error)
  }
}
const handleVoteDeadline = async () => {
  try {
    let tx = await contract.votingDeadline()
    setVoteDeadlne(tx)
    console.log(tx);
    // alert(wait.transactionHash)
  } catch (error) {
```

```
      alert(error)
    }
  }
  const handleElecName = async () => {
    try {
      let tx = await contract.electionName()
      setElection(tx)
      console.log(tx);
      // alert(wait.transactionHash)
    } 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" }}>Ballot Box 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={handleCandidateIndex} type="number" placeholder="Candidate Index"
value={CandidateIndex} /> <br />

      <Button onClick={handleCastVote} style={{ marginTop: "10px" }}
variant="primary">Cast Vote</Button>

      </div>

     </Col>

     <Col style={{ marginRight: "100px" }}>

        <div>

          <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleVoterBiometricData} type="string" placeholder="Vote Encripted
data" value={VoterData} /> <br />

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

        </div>

     </Col>

   </Row>

   <Row style={{marginTop:"100px"}}>

            <Col style={{ marginRight: "100px" }}>

              <div>
```

```
            <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleCandidateIndexs} type="number" placeholder="Candidate Index"
value={CandidateIndexed} /> <br />

              <Button onClick={handleCandidate} style={{ marginTop: "10px" }}
variant="primary">  Get transaction Count</Button>

            {CandidatesData ? CandidatesData?.map(e => <p>{e.toString()}</p>) :
<p></p>

             }

           </div>

         </Col>

         <Col style={{ marginRight: "100px" }}>

           <div>

            <Button onClick={handleRegdeadline} style={{ marginTop: "10px" }}
variant="primary">Registration deadline</Button>

            {RegDeadline ? <p>{RegDeadline.toString()}</p> : <p></p>}

           </div>

         </Col>

    </Row>

      <Row style={{ marginTop: "50px" }}>

        <Col style={{ marginRight: "100px" }}>

          <div>

            <Button onClick={handleVoteDeadline} style={{ marginTop: "10px" }}
variant="primary">Voting deadline</Button>

            {VoteDeadlne ? <p>{VoteDeadlne.toString()}</p> : <p></p>}

          </div>

        </Col>

        <Col style={{ marginRight: "100px" }}>

          <div>

            <Button onClick={handleElecName} style={{ marginTop: "10px" }}
variant="primary">Election Name</Button>
```

```
                {Election ? <p>{Election.toString()}</p> : <p></p>}
              </div>
            </Col>
  </Row>
  </Container>
 </div>
)
}
export default Home;
```

**Connector Code**

```
const { ethers } = require("ethers");
const abi = [
 {
  "inputs": [
   {
    "internalType": "string",
    "name": "_electionName",
    "type": "string"
   },
   {
    "internalType": "uint256",
    "name": "_registrationDeadline",
    "type": "uint256"
   },
   {
```

```
   "internalType": "uint256",
   "name": "_votingDeadline",
   "type": "uint256"
  },
  {
   "internalType": "string[]",
   "name": "_candidateNames",
   "type": "string[]"
  }
 ],
 "stateMutability": "nonpayable",
 "type": "constructor"
},
{
 "anonymous": false,
 "inputs": [
  {
   "indexed": true,
   "internalType": "address",
   "name": "voter",
   "type": "address"
  },
  {
   "indexed": false,
   "internalType": "uint256",
   "name": "candidateIndex",
   "type": "uint256"}
```

```
 ],
 "name": "VoteCast",
 "type": "event"
},
{
 "inputs": [
  {
   "internalType": "uint256",
   "name": "",
   "type": "uint256"
  }
 ],
 "name": "candidates",
 "outputs": [
  {
   "internalType": "string",
   "name": "name",
   "type": "string"
  },
  {
   "internalType": "uint256",
   "name": "voteCount",
   "type": "uint256"
  }
 ],
 "stateMutability": "view",
 "type": "function"
```

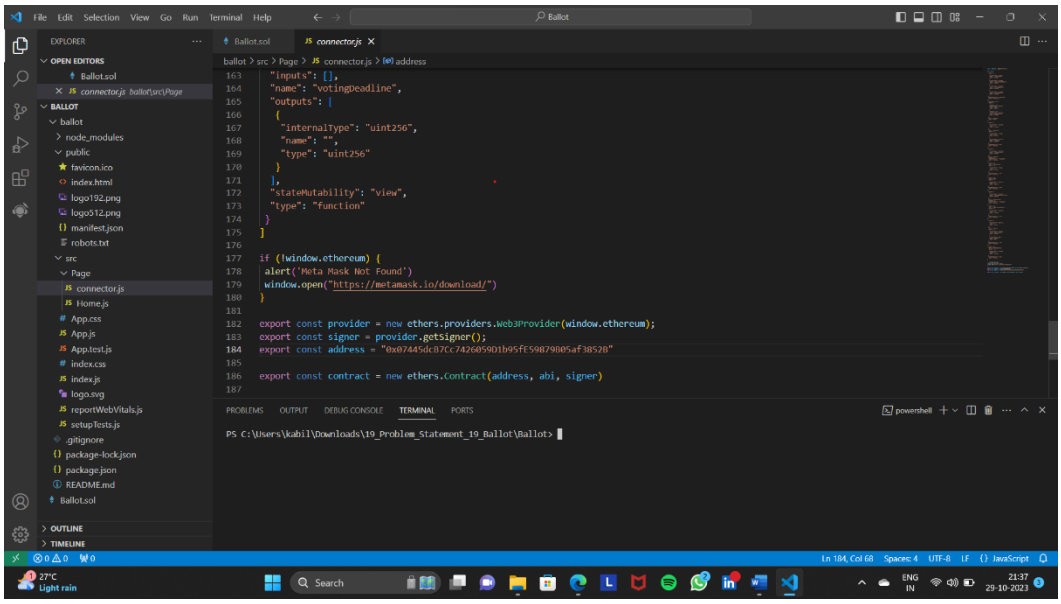
```
},
{
 "inputs": [
  {
   "internalType": "uint256",
   "name": "_candidateIndex",
   "type": "uint256"
  }
 ],
 "name": "castVote",
 "outputs": [],
 "stateMutability": "nonpayable",
 "type": "function"
},
{
 "inputs": [],
 "name": "electionName",
 "outputs": [
  {
   "internalType": "string",
   "name": "",
   "type": "string"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
```

```
{
 "inputs": [],
 "name": "owner",
 "outputs": [
  {
   "internalType": "address",
   "name": "",
   "type": "address"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [
  {
   "internalType": "bytes32",
   "name": "_encryptedBiometricData",
   "type": "bytes32"
  }
 ],
 "name": "registerVoter",
 "outputs": [],
 "stateMutability": "nonpayable",
 "type": "function"
},
{
```

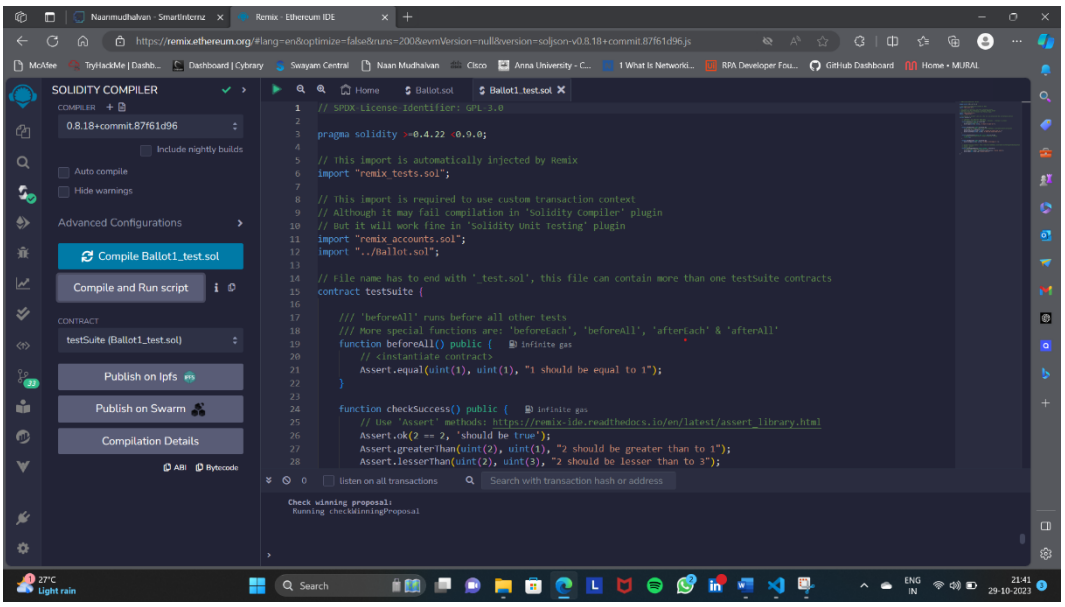
```
"inputs": [],
"name": "registrationDeadline",
"outputs": [
 {
  "internalType": "uint256",
  "name": "",
  "type": "uint256"
 }
],
"stateMutability": "view",
"type": "function"
},
{
"inputs": [
 {
  "internalType": "address",
  "name": "",
  "type": "address"
 }
],
"name": "voters",
"outputs": [
 {
  "internalType": "bytes32",
  "name": "biometricData",
  "type": "bytes32"
 },
```

```
    {
     "internalType": "bool",
     "name": "hasVoted",
     "type": "bool"
    }
   ],
   "stateMutability": "view",
   "type": "function"
  },
  {
   "inputs": [],
   "name": "votingDeadline",
   "outputs": [
    {
     "internalType": "uint256",
     "name": "",
     "type": "uint256"
    }
   ],
   "stateMutability": "view",
   "type": "function"
  }
]
if (!window.ethereum) {
 alert('Meta Mask Not Found')
 window.open("https://metamask.io/download/")
}
```

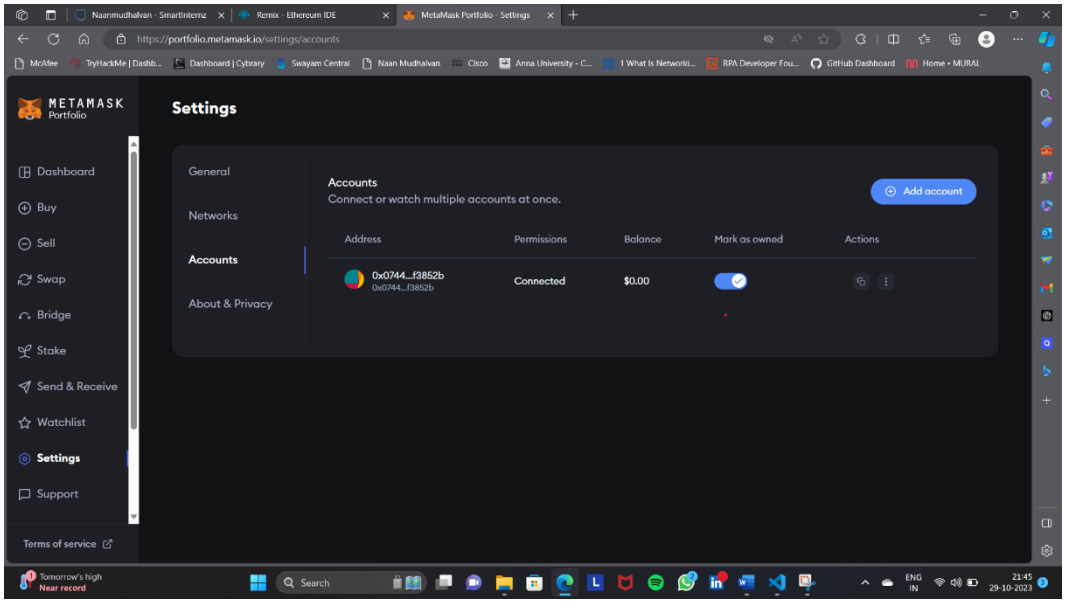
```
export const provider = new ethers.providers.Web3Provider(window.ethereum);
export const signer = provider.getSigner();
export const address = "0x07445dcB7Cc7426059D1b95fE59879B05af3852B"
export const contract = new ethers.Contract(address, abi, signer)
```




Ballot code in VS Code




Code in Remix while compile




Metamask Account

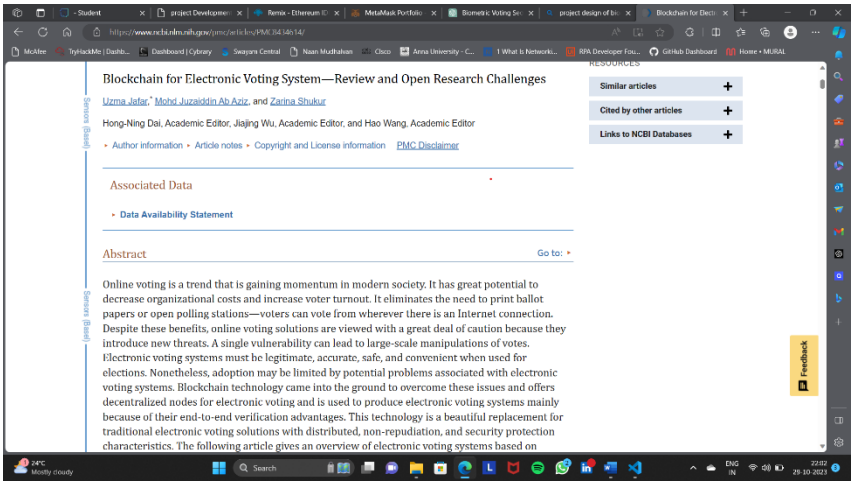
## PERFORMANCE TESTING

Performance testing of the Biometric Security System for a Voting Platform is crucial to ensure that the system operates efficiently, reliably, and can handle the expected load during elections. Performance testing ensures that the Biometric Security System can handle the demands of real-world elections, providing a secure and efficient voting experience for all users.

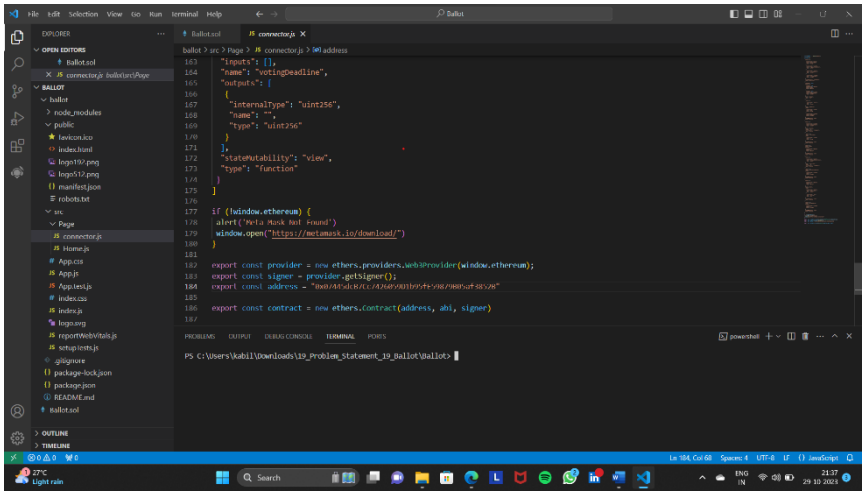
### Performance Metrics

- **Response Time:**
  **Metric:** The time it takes for the system to respond to a user request.
  **Objective:** Ensure low and consistent response times to provide a smooth and responsive user experience during voter authentication and other interactions.
- **Throughput:**
  **Metric:** The number of transactions or operations the system can handle per unit of time.
  **Objective:** Measure the system's capacity to process a high volume of voter authentication requests and other transactions during peak voting periods.
- **Concurrency:**
  **Metric:** The number of simultaneous users or connections the system can handle.
  **Objective:** Evaluate the system's ability to support multiple users accessing the platform concurrently without performance degradation.

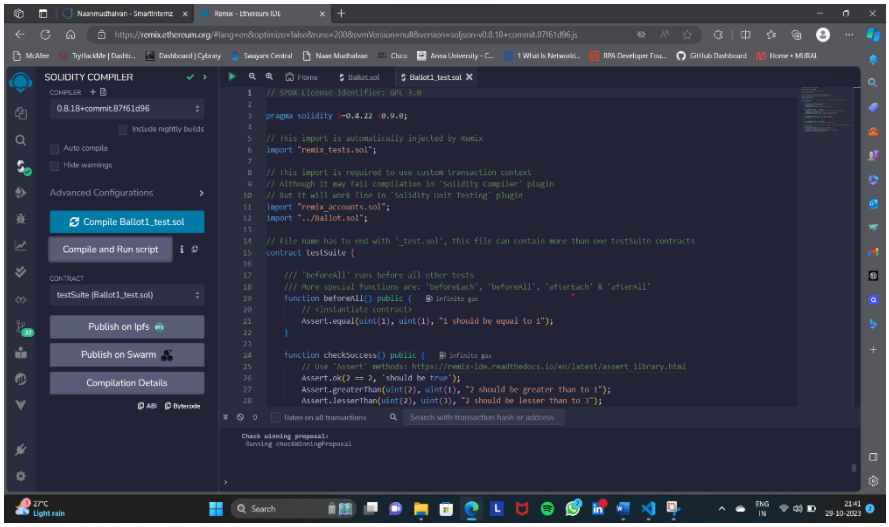
- **Scalability:**
  **Metric:** The system's ability to handle increased load by adding resources.
  **Objective:** Assess how well the system scales vertically (adding resources to a single server) and horizontally (adding more servers) to meet growing demands.
- **Error Rate:**
  **Metric:** The percentage of transactions that result in errors.
  **Objective:** Minimize errors during voter authentication and other critical processes to ensure the accuracy and reliability of the voting system.
- **Database Performance:**
  **Metric:** Database response time and throughput for read and write operations.
  **Objective:** Optimize database performance to ensure efficient retrieval and storage of voter information and voting records.
- **Network Latency:**
  **Metric:** The time it takes for data to travel between the client and server.
  **Objective:** Minimize network latency to enhance the speed and responsiveness of the biometric authentication process.
- **Transaction Success Rate:**
  **Metric:** The percentage of successful transactions.
  **Objective:** Maintain a high success rate for voter authentication and other transactions to prevent service disruptions and ensure a positive user experience.
- **Load Balancing Efficiency:**
  **Metric:** The effectiveness of distributing load across multiple servers.
  **Objective:** Optimize load balancing mechanisms to evenly distribute traffic and prevent overloading of specific servers.
- **Redundancy and Failover Effectiveness:**
  **Metric:** The system's ability to switch to redundant components in case of failure.
  **Objective:** Verify that redundancy and failover mechanisms work seamlessly to ensure continuous operation and minimize the impact of component failures.
- **Security Performance:**
  **Metric:** Performance impact of security measures (e.g., encryption, authentication).
  **Objective:** Balance robust security measures with minimal impact on system performance, ensuring a secure voting environment.



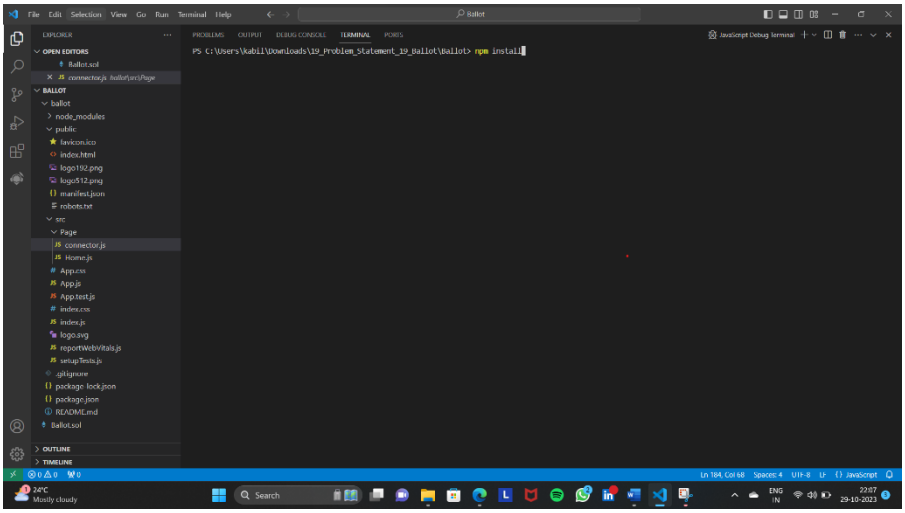

1. Information gathering



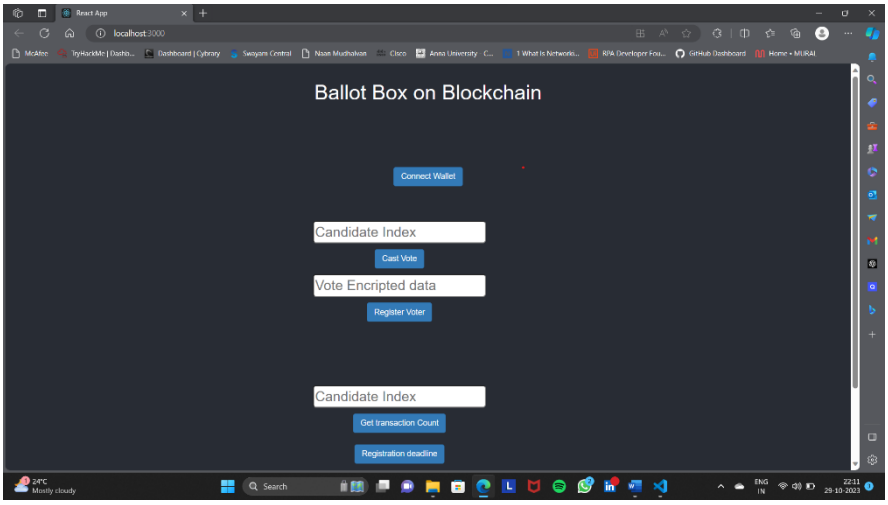

2. Open to vs code




3. Remix Ide platform explorting

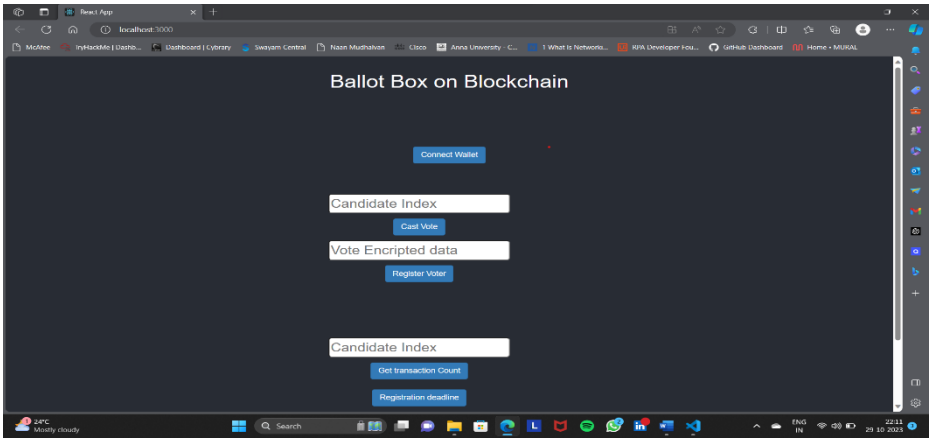


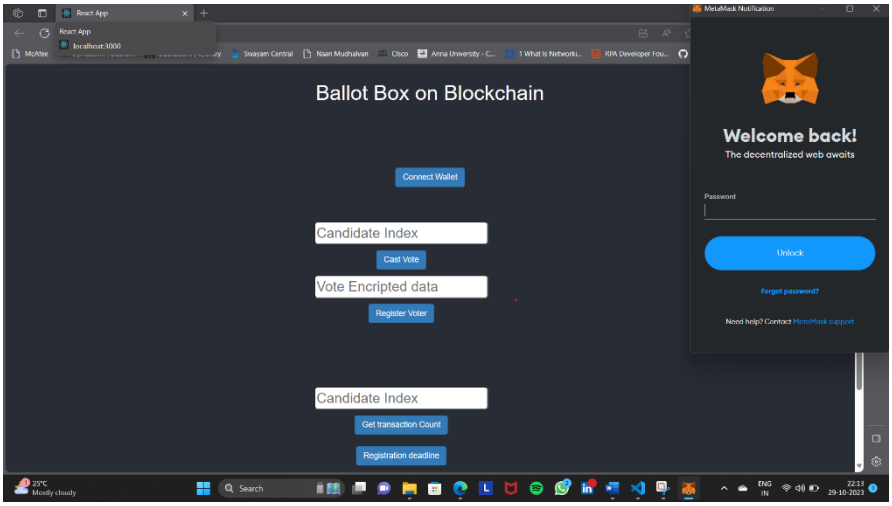

4. Open file and done installation and code started

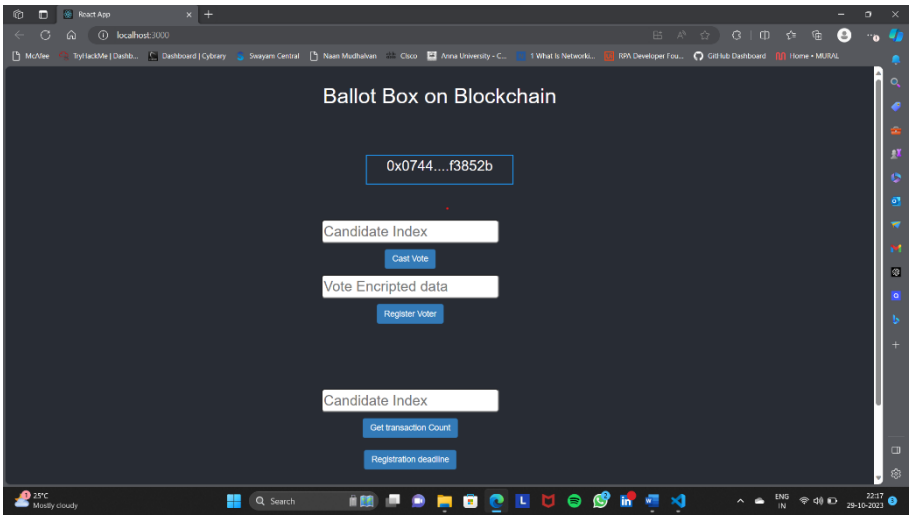


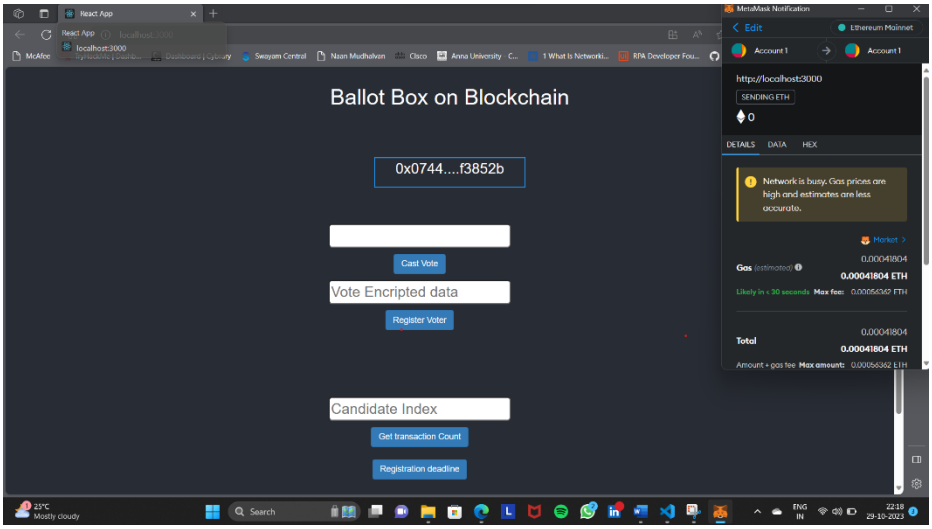

5. LOCALHOST IP ADDRESS

## RESULTS




Ballot Box on Blockchain
Connect Wallet
Candidate Index
Cast Vote
Vote Encripted data
Register Voter
Candidate Index
Get transaction Count
Registration deadline
Welcome back!
The decentralized web awaits
Password
Unlock
Forgot password?


Ballot Box on Blockchain
0x0744....f3852b
Candidate Index
Cast Vote
Vote Encripted data
Register Voter
Candidate Index
Get transaction Count
Registration deadline


Ballot Box on Blockchain
0x0744....f3852b
Cast Vote
Vote Encripted data
Register Voter
Candidate Index
Get transaction Count
Registration deadline
Edit
Ethereum Mainnet
Account 1
Account 1
http://localhost:3000
SENDING ETH
0
DETAILS
DATA
HEX
Network is busy. Gas prices are high and estimates are less accurate.
Gas
0.00041804
0.00041804 ETH
Total
0.00041804
0.00041804 ETH

## ADVANTAGES & DISADVANTAGES

### Advantages:

- **Enhanced Security:** Biometric authentication provides a high level of security by verifying voters based on unique physical or behavioral traits, reducing the risk of identity fraud.
- **Accurate Identification:** Biometric systems offer accurate and reliable identification, minimizing the chances of impersonation and ensuring that each vote is attributed to the correct individual.
- **Reduced Voter Fraud:** By relying on biometric data, the system mitigates the risk of common voting fraud tactics, such as multiple voting or using false identities.
- **Increased Transparency:** Integration with blockchain technology can enhance transparency and trust by creating an immutable and publicly accessible ledger of votes.
- **User Convenience:** Biometric authentication is user-friendly and convenient for voters, eliminating the need for traditional identification methods like ID cards or passwords.
- **Fast and Efficient:** Biometric authentication processes are generally quick and efficient, allowing for swift voter verification during the election process.

### Disadvantages:

- **Biometric Data Privacy Concerns:** Storing and handling biometric data raises privacy concerns. Ensuring robust security measures and compliance with data protection regulations is crucial.
- **Technical Challenges:** Technical challenges, such as errors in biometric matching or sensor malfunctions, can impact the system's accuracy and reliability.
- **Initial Implementation Costs:** Implementing a biometric security system involves significant upfront costs for acquiring biometric sensors, developing the system, and ensuring compliance with security standards.
- **Integration Complexities:** Integrating biometric systems with existing voting platforms can be complex and may require significant adjustments to ensure seamless operation.
- **Voter Acceptance and Trust:** Some voters may have reservations about the use of biometrics, citing concerns about privacy, security, or a general mistrust of new technologies.
- **Vulnerability to Spoofing:** Despite advancements in biometric technology, there remains a potential vulnerability to spoofing or mimicking biometric

traits, emphasizing the need for continuous improvement in biometric algorithms.

- **Limited Accessibility:** Biometric systems may pose challenges for individuals with certain disabilities or those whose biometric features are difficult to capture accurately.
- **Legal and Ethical Issues:** Legal and ethical concerns may arise regarding the collection, storage, and use of biometric data. Clear guidelines and compliance with regulations are essential.

## CONCLUSION

In conclusion, the integration of a Biometric Security System into a Voting Platform represents a significant leap forward in enhancing the integrity and security of the democratic process. The advantages, such as heightened security, accurate identification, and reduced voter fraud, underscore the potential for a more transparent and trustworthy electoral system.

However, careful consideration must be given to the challenges and concerns associated with biometric technology. Privacy issues, technical complexities, and the need for substantial initial investments highlight the importance of a well-thought-out implementation strategy. Striking a balance between the advantages of biometric security and the mitigation of its disadvantages is crucial for ensuring widespread acceptance and maintaining the democratic principles of accessibility and inclusivity.

In navigating the path forward, collaboration with stakeholders, adherence to data protection regulations, and a commitment to user education and awareness will be key. The journey towards a more secure and reliable voting system requires a thoughtful and adaptive approach, one that harnesses the benefits of biometric technology while safeguarding the fundamental principles of democratic participation.

## FUTURE SCOPE

The future scope of Biometric Security Systems for Voting Platforms envisions a landscape of heightened security, efficiency, and user-centricity. Advancements in biometric technologies, including AI and machine learning integration, promise more accurate and adaptable systems. Innovations in data encryption, decentralized identity solutions, and privacy-preserving techniques anticipate a future where individual control over biometric data is paramount. Quantum-resistant algorithms and multi-factor authentication enhancements aim to fortify the resilience of these systems against evolving threats. Additionally, efforts toward global standardization,

interoperability, and continuous research underscore a commitment to maintaining the highest standards of security in electoral processes. Public awareness and education campaigns will play a pivotal role in fostering trust and acceptance of these evolving technologies, ensuring that the future of Biometric Security Systems aligns with democratic principles of transparency, inclusivity, and integrity.

## APPENDIX

### Source Code

**Ballot.sol**

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

contract BallotBox {
    // Define the owner of the contract (election authority).
    address public owner;

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

    // Define the structure of a candidate.
    struct Candidate {
        string name;
        uint256 voteCount;
    }

    // Define the election parameters.
```

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

// Store the list of candidates.
Candidate[] public candidates;

// Store the mapping of voters.
mapping(address => Voter) public voters;

// Event to announce when a vote is cast.
event VoteCast(address indexed voter, uint256 candidateIndex);

// Modifiers for access control.
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 to initialize the contract.
```

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

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

// Function to register a voter and store their encrypted biometric data.
function registerVoter(bytes32 _encryptedBiometricData) public canVote {
    voters[msg.sender] = Voter({
        biometricData: _encryptedBiometricData,
        hasVoted: false
    });
}
```

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

        // Mark the voter as having voted.
        voters[msg.sender].hasVoted = true;

        // Increment the candidate's vote count.
        candidates[_candidateIndex].voteCount++;

        // Emit a VoteCast event.
        emit VoteCast(msg.sender, _candidateIndex);
    }
}
```

**Home Page.js**

```
import React, { useState } from "react";
import { Button, Container, Row, Col } from 'react-bootstrap';
import 'bootstrap/dist/css/bootstrap.css';
import { contract } from "./connector";
function Home() {
  const [Wallet, setWallet] = useState("");
  const [CandidateIndex, setCandidateIndex] = useState("");
  const [VoterData, setVoterData] = useState("");
  const [CandidateIndexed, setCandidateIndexed] = useState("");
```

```
  const [CandidatesData, setCandidatesData] = useState("");
  const [RegDeadline, setRegDeadlne] = useState("");
  const [VoteDeadlne, setVoteDeadlne] = useState("");
  const [Election, setElection] = useState("");
  const handleCandidateIndex = (e) => {
    setCandidateIndex(e.target.value)
}
  const handleCastVote = async () => {
    try {
      let tx = await contract.castVote(CandidateIndex.toString())
      let wait = await tx.wait()
      console.log(wait);
      alert(wait.transactionHash)
    } catch (error) {
      alert(error)
    }
  }
  const handleVoterBiometricData = (e) => {
    setVoterData(e.target.value)
  }
 const handleRegisterVoter = async () => {
    try {
      let tx = await contract.registerVoter(VoterData)
      let wait = await tx.wait()
      console.log(wait)
      alert(wait.transactionHash)
    } catch (error) {
```

```
      alert(error)
    }
  }
  const handleCandidateIndexs = (e) => {
    setCandidateIndexed(e.target.value)
  }
  const handleCandidate = async () => {
    try {
      let tx = await contract.candidates(CandidateIndexed.toString())
      setCandidatesData(tx)
      console.log(tx);
      // alert(wait.transactionHash)
    } catch (error) {
      alert(error)
    }
  }
  const handleRegdeadline = async () => {
    try {
      let tx = await contract.registrationDeadline()
      setRegDeadlne(tx)
      console.log(tx);
      // alert(wait.transactionHash)
    } catch (error) {
      alert(error)
    }
  }
  const handleVoteDeadline = async () => {
```

```
    try {
      let tx = await contract.votingDeadline()
      setVoteDeadlne(tx)
      console.log(tx);
      // alert(wait.transactionHash)
    } catch (error) {
     alert(error)
    }
  }
  const handleElecName = async () => {
    try {
      let tx = await contract.electionName()
      setElection(tx)
      console.log(tx);
      // alert(wait.transactionHash)
    } 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" }}>Ballot Box 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={handleCandidateIndex} type="number" placeholder="Candidate Index"
value={CandidateIndex} /> <br />
      <Button onClick={handleCastVote} style={{ marginTop: "10px" }}
variant="primary">Cast Vote</Button>
      </div>
     </Col>
     <Col style={{ marginRight: "100px" }}>
        <div>
          <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleVoterBiometricData} type="string" placeholder="Vote Encripted
data" value={VoterData} /> <br />
          <Button onClick={handleRegisterVoter} style={{ marginTop: "10px" }}
variant="primary">Register Voter</Button>
```

```
        </div>
      </Col>
    </Row>
    <Row style={{marginTop:"100px"}}>
            <Col style={{ marginRight: "100px" }}>
              <div>
               <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleCandidateIndexs} type="number" placeholder="Candidate Index"
value={CandidateIndexed} /> <br />
                <Button onClick={handleCandidate} style={{ marginTop: "10px" }}
variant="primary">  Get transaction Count</Button>
               {CandidatesData ? CandidatesData?.map(e => <p>{e.toString()}</p>) :
<p></p>
                }
              </div>
            </Col>
            <Col style={{ marginRight: "100px" }}>
              <div>
               <Button onClick={handleRegdeadline} style={{ marginTop: "10px" }}
variant="primary">Registration deadline</Button>
               {RegDeadline ? <p>{RegDeadline.toString()}</p> : <p></p>}
              </div>
            </Col>
      </Row>
        <Row style={{ marginTop: "50px" }}>
          <Col style={{ marginRight: "100px" }}>
            <div>
              <Button onClick={handleVoteDeadline} style={{ marginTop: "10px" }}
variant="primary">Voting deadline</Button>
```

```
            {VoteDeadlne ? <p>{VoteDeadlne.toString()}</p> : <p></p>}
          </div>
        </Col>
        <Col style={{ marginRight: "100px" }}>
          <div>
            <Button onClick={handleElecName} style={{ marginTop: "10px" }}
variant="primary">Election Name</Button>
            {Election ? <p>{Election.toString()}</p> : <p></p>}
          </div>
        </Col>
  </Row>
  </Container>
 </div>
 )
}
export default Home;
```

**Connector.js code**

```
const { ethers } = require("ethers");
const abi = [
 {
  "inputs": [
   {
    "internalType": "string",
    "name": "_electionName",
    "type": "string"
   },
```

```
 {
  "internalType": "uint256",
  "name": "_registrationDeadline",
  "type": "uint256"
 },
 {
  "internalType": "uint256",
  "name": "_votingDeadline",
  "type": "uint256"
 },
 {
  "internalType": "string[]",
  "name": "_candidateNames",
  "type": "string[]"
 }
],
"stateMutability": "nonpayable",
"type": "constructor"
},
{
"anonymous": false,
"inputs": [
 {
  "indexed": true,
  "internalType": "address",
  "name": "voter",
  "type": "address"
```

```
  },
  {
   "indexed": false,
   "internalType": "uint256",
   "name": "candidateIndex",
   "type": "uint256"}
 ],
 "name": "VoteCast",
 "type": "event"
},
{
 "inputs": [
  {
   "internalType": "uint256",
   "name": "",
   "type": "uint256"
  }
 ],
 "name": "candidates",
 "outputs": [
  {
   "internalType": "string",
   "name": "name",
   "type": "string"
  },
  {
   "internalType": "uint256",
```

```
    "name": "voteCount",
    "type": "uint256"
   }
  ],
  "stateMutability": "view",
  "type": "function"
 },
 {
  "inputs": [
   {
    "internalType": "uint256",
    "name": "_candidateIndex",
    "type": "uint256"
   }
  ],
  "name": "castVote",
  "outputs": [],
  "stateMutability": "nonpayable",
  "type": "function"
 },
 {
  "inputs": [],
  "name": "electionName",
  "outputs": [
   {
    "internalType": "string",
    "name": "",
```

```
  "type": "string"
 }
],
"stateMutability": "view",
"type": "function"
},
{
"inputs": [],
"name": "owner",
"outputs": [
 {
  "internalType": "address",
  "name": "",
  "type": "address"
 }
],
"stateMutability": "view",
"type": "function"
},
{
"inputs": [
 {
  "internalType": "bytes32",
  "name": "_encryptedBiometricData",
  "type": "bytes32"
 }
],
```

```
 "name": "registerVoter",
 "outputs": [],
 "stateMutability": "nonpayable",
 "type": "function"
},
{
 "inputs": [],
 "name": "registrationDeadline",
 "outputs": [
  {
   "internalType": "uint256",
   "name": "",
   "type": "uint256"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [
  {
   "internalType": "address",
   "name": "",
   "type": "address"
  }
 ],
 "name": "voters",
```

```
 "outputs": [
  {
   "internalType": "bytes32",
   "name": "biometricData",
   "type": "bytes32"
  },
  {
   "internalType": "bool",
   "name": "hasVoted",
   "type": "bool"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [],
 "name": "votingDeadline",
 "outputs": [
  {
   "internalType": "uint256",
   "name": "",
   "type": "uint256"
  }
 ],
 "stateMutability": "view",
 "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 = "0x07445dcB7Cc7426059D1b95fE59879B05af3852B"
export const contract = new ethers.Contract(address, abi, signer)
```

**Github Link:** https://github.com/kabilesh11/NM2023TMID07060

**Video Demo Link:** https://drive.google.com/file/d/18sbVSqb0SJf5MJunGBB8nu94Ui8mHp4H/view?usp=sharing