**A Project Report**

**On**

**‘‘Virtual Voting System’’**

submitted for partial fulfillment of the requirements

for the award of the degree of

Bachelor of Technology

in

Computer Science

**Submitted by**

Surya Pratap Singh (2000290120169)

Utkarsh Mishra (2000290120180)

Adrika Tripathi (2000290120015)

**Under supervision of**

Prof. Akanksha



**KIET Group of Institutions, Ghaziabad**

**Dr. A.P.J. Abdul Kalam Technical University, Lucknow**

**May 2024**

**DECLARATION**

I/We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature

Name: - Surya Pratap Singh, Utkarsh Mishra, Adrika Tripathi

Roll No.: - 2000290120169, 2000290120180, 2000290120015

Date: - 11-03-2024

## CERTIFICATE

This is to certify that Project Report entitled **“Virtual Voting System”** which is submitted by **Surya Pratap Singh, Utkarsh Mishra, Adrika Tripathi** in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science of Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

.

**Date:** 11-03-2024

**Supervisor:**

Akanksha

(Assistant Professor- Department of Computer Science)

**ACKNOWLEDGEMENT**

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe a special debt of gratitude to Professor (Akanksha), Department of Computer Science, KIET, Ghaziabad, for his constant support and guidance throughout our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Dr. Ajay Kumar Shrivastava, Head of the Department of Computer Science, KIET, Ghaziabad, for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all the faculty members of the department for their kind assistance and cooperation during the development of our project.

Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Date : 11-03-2024

|  |  |  |
| --- | --- | --- |
| Surya Pratap Singh | Utkarsh Mishra | Adrika Tripathi` |
| 2000290120169 | 2000290120180 | 2000290120015 |

**ABSTRACT**

The organization of elections by governments is extremely important to people in today's world. Elections play an important role in the governance of a country or organization, or as it is sometimes said, it is the factor that determines the fate of any country. Even the biggest democracies in the world, including India and the United States, nevertheless have unreliable voting procedures. The biggest problems in the current electoral system are vote rigging, EVM hacking, election manipulation and vote hijacking.

Blockchain is a new, decentralized and decentralized technology that has the potential to improve many elements in many sectors. The problem of the current electronic voting system can be solved by extending it with blockchain technology. The blockchain with smart contracts stands out as a strong contender to be used in the creation of safer, more affordable, more secure, more transparent, and simpler electronic voting systems.We developed and tested a virtual voting application for the Ethereum network as a smart contract using the Solidity programming language and the Ethereum platform. Because of its widespread use, reliability, and abundance of smart contract logic, Ethereum and its network rank among the finest. Duplicate vote prevention, full transparency, and privacy protection for participants are all ensured via a secure electronic voting method. The implementation of the virtual voting system involves the development of smart contracts to manage the voting process, ensuring that only eligible voters can participate and that votes are counted accurately. Additionally, mechanisms for voter authentication and verification are integrated into the system to prevent fraudulent activities.

Through the utilization of blockchain technology, the proposed virtual voting system offers a reliable and efficient solution for conducting elections, facilitating remote participation while maintaining the integrity and transparency of the electoral process. However, further research and testing are required to address scalability issues and ensure widespread adoption of this innovative approach to voting.

| TABLE OF CONTENTS | Page No. |
| --- | --- |
| DECLARATION……………………………………………………………………. | ii |
| CERTIFICATE……………………………………………………………………… | iii |
| ACKNOWLEDGEMENTS…………………………………………………………. | iv |
| ABSTRACT……………………………………………………………………….... | v |
| LIST OF FIGURES…………………………………………………………………. | ix |
| LIST OF TABLES………………………………………………….......................... | xi |
| LIST OF ABBREVIATIONS………………………………………………………. | xii |
|  |  |
| 1. CHAPTER 1: INTRODUCTION    1. INTRODUCTION TO PROJECT    2. PROJECT CATEGORY (INTERNET BASED, APPLICATION OR SYSTEM DEVELOPMENT, RESEARCH BASED, INDUSTRY AUTOMATION, NETWORK OR SYSTEM ADMINISTRATION)    3. OBJECTIVES    4. PROBLEM FORMULATION    5. PROPOSED SYSTEM    6. UNIQUE FEATURES OF THE SYSTEM 2. CHAPTER 2. REQUIREMENT ANALYSIS & SYSTEM SPECIFICATION    1. FEASIBILITY STUDY (TECHNICAL, ECONOMICAL, OPERATIONAL)    2. SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT WHICH MUST INCLUDE THE FOLLOWING: (DATA REQUIREMENT, FUNCTIONAL REQUIREMENT, PERFORMANCE REQUIREMENT, MAINTAINABILITY REQUIREMENT, SECURITY REQUIREMENT)    3. SDLC MODEL TO BE USED 3. . CHAPTER 3. SYSTEM DESIGN    1. DETAIL DESIGN    2. SYSTEM DESIGN USING DFD LEVEL 0 AND LEVEL 1    3. USE CASE DIAGRAM    4. DATABASE DESIGN    5. ER DIAGRAMS 4. CHAPTER 4. IMPLEMENTATION, TESTING, AND MAINTENANCE    1. INTRODUCTION TO LANGUAGES, TOOLS AND TECHNOLOGIES USED FOR IMPLEMENTATION    2. TESTING TECHNIQUES AND TEST CASES USED 5. CHAPTER 5. RESULTS AND DISCUSSIONS    1. USER INTERFACE REPRESENTATION (OF RESPECTIVE PROJECT)    2. BRIEF DESCRIPTION OF VARIOUS MODULES OF THE SYSTEM    3. SNAPSHOTS OF SYSTEM WITH BRIEF DETAIL OF EACH    4. BACK-ENDS REPRESENTATION (DATABASE TO BE USED)    5. SNAPSHOTS OF DATABASE TABLES WITH BRIEF DESCRIPTION 6. CHAPTER 6. CONCLUSION AND FUTURE SCOPE   REFERENCES (REFERENCES SHOULD BE IN IEEE STANDARD) | Page No. |

**LIST OF FIGURES**

| **Figure No.** | **Description** | **Page No.** |
| --- | --- | --- |
| 1.1 | Categorization of routing protocol | 2 |
| 2.1 | Three models | 9 |
| 3.1 | Resolving failed links in DSDV | 12 |

**LIST OF TABLES**

| **Table. No.** | **Description** | **Page No.** |
| --- | --- | --- |
| 1.1 | General Simulation parameters | 22 |

**LIST OF ABBREVIATIONS**

NAM Network Animator

MANET Mobile Ad-hoc Network

DSDV Destination Sequenced Distance Vector

DSR Dynamic Sequence Resource

AODV Ad-hoc On-Demand Vector

OSI Open System Interconnections

TCP/IP Transmission Control Protocol/Internet Protocol

Pdf Packet Drop Fraction

GSR Global State Routing

**CHAPTER 1**

**INTRODUCTION**

**1.1 INTRODUCTION**

In an era characterized by unprecedented technological advancements, the traditional methods of conducting elections have come under increasing scrutiny. The digital age has ushered in the potential for innovation in the electoral process, promising greater transparency, security, and accessibility. One groundbreaking development that has garnered significant attention is the integration of blockchain technology into the realm of voting, giving rise to what is commonly referred to as "virtual voting”. The concept of virtual voting using blockchain technology is both revolutionary and disruptive. It offers the promise of a democratic system that is more secure, transparent, and efficient. This novel approach combines the security features of blockchain, such as decentralization and cryptographic protection, with the accessibility and convenience of a virtual voting platform. As a result, it has the potential to address longstanding concerns related to electoral integrity, voter fraud, and accessibility.

The virtual voting system proposed in this project embodies the principles of decentralization, transparency, security, anonymity, and verifiability. Through the utilization of blockchain's distributed ledger technology, every aspect of the voting process—from voter registration to ballot casting and tallying—is recorded in a tamper-proof and transparent manner, thereby mitigating the risks associated with fraud and manipulation.

The implementation of this virtual voting system involves the development of smart contracts to automate and enforce the rules governing the electoral process, as well as the integration of robust security measures to safeguard against potential threats. While challenges such as scalability and usability remain to be addressed, the potential benefits of adopting blockchain-based voting systems are vast, offering a pathway towards more inclusive, trustworthy, and democratic elections.

**1.2 PROJECT CATEGORY**

**Internet Based**: The project involves the development of a virtual voting system that utilizes blockchain technology. This system would likely be accessed and interacted with over the internet, allowing voters to participate remotely from any location with internet connectivity. The use of blockchain technology inherently involves distributed networks, and its implementation for voting purposes would require internet connectivity for participants to access the voting interface, submit their votes securely, and verify the integrity of the process.

While aspects of "Research Based" could also apply, as the project involves innovation and exploration of new technologies (such as blockchain), the primary focus seems to be on developing an application or system (the virtual voting system) that can be accessed and utilized over the internet.

Other categories such as "Application or System Development" could also be relevant, as the project entails the development of a specific application or system (the virtual voting platform). However, the emphasis on internet-based access and interaction distinguishes it as primarily fitting into the "Internet Based" category.

**1.3 OBJECTIVE**

**Enhance Electoral Integrity**: The primary goal of the project is to develop a virtual voting system that significantly enhances the integrity of electoral processes. By leveraging blockchain technology, the system aims to minimize the risk of fraud, manipulation, and other irregularities that may compromise the fairness and legitimacy of elections.

**Ensure Transparency and Accountability**: Another objective is to promote transparency and accountability in the electoral process. Through the use of blockchain's transparent and immutable ledger, the system aims to provide a verifiable record of all voting activities, enabling stakeholders to audit and verify the integrity of the election results.

**Facilitate Remote Participation**: The project seeks to enable remote participation in elections by providing a secure and accessible platform for voters to cast their ballots from any location with internet connectivity. This objective aims to overcome barriers such as geographical distance, mobility issues, and other constraints that may hinder voter participation.

**Protect Voter Privacy:** The project aims to safeguard the privacy and anonymity of voters while ensuring the integrity of the voting process. By employing cryptographic techniques and decentralized architecture, the system aims to protect sensitive voter information and prevent unauthorized access to voting data.

**Promote Trust in Democratic Processes**: Ultimately, the overarching objective is to promote trust and confidence in democratic processes. By offering a secure, transparent, and inclusive voting system, the project seeks to strengthen public trust in the electoral system and uphold the principles of democracy.

**Research and Innovation:** Additionally, the project aims to contribute to the advancement of research and innovation in the field of blockchain technology and e-governance. By exploring the potential applications of blockchain in electoral processes, the project seeks to drive forward-thinking solutions for addressing societal challenges related to governance and democracy.

Overall, the objectives of the project align with the broader goals of promoting electoral integrity, transparency, accessibility, and trust in democratic processes through the innovative application of blockchain technology.

**1.4 PROBLEM FORMULATION**

**1. Existing Challenges in Electoral Systems:**

* Traditional electoral systems are susceptible to various challenges such as fraud, manipulation, and lack of transparency.
* Geographical barriers and logistical constraints often limit voter participation, particularly in remote or inaccessible areas.
* Privacy concerns and security risks threaten the integrity of the voting process, undermining public trust in democratic institutions.

**2. Need for Secure and Transparent Voting Solutions:**

* There is a pressing need for innovative voting solutions that can address the shortcomings of traditional electoral systems.
* Ensuring the security, transparency, and integrity of the voting process is paramount to upholding democratic principles and fostering public trust in electoral outcomes.
* Remote participation in elections has become increasingly important, especially in the context of global mobility and digital connectivity.

**3. Opportunities Offered by Blockchain Technology:**

* Blockchain technology offers unique features such as decentralization, transparency, immutability, and cryptographic security.
* Leveraging blockchain for voting systems has the potential to mitigate existing challenges by providing a tamper-proof and transparent platform for conducting elections.
* Blockchain's decentralized architecture enables trustless interactions among participants, eliminating the need for intermediaries and reducing the risk of manipulation

**1.5 PROPOSED SYSTEM**

**1. Blockchain Infrastructure:**

* Implement a decentralized blockchain network that serves as the foundation for the virtual voting system.
* Choose a suitable blockchain platform (e.g., Ethereum, Hyperledger Fabric) based on factors such as scalability, security, and ease of integration.
* Deploy smart contracts to manage the voting process, including voter registration, ballot casting, and result tallying, ensuring transparency and immutability of data.

**2. User Authentication and Authorization:**

* Develop a robust authentication system to verify the identity and eligibility of voters before allowing them to participate in the election.
* Implement cryptographic techniques such as digital signatures to ensure the integrity and authenticity of voter credentials.
* Enable multi-factor authentication methods to enhance security and prevent unauthorized access to the voting system.

**3. Voter Registration and Verification**:

* Design a user-friendly interface for voter registration, allowing individuals to securely register their identity and eligibility criteria.
* Integrate verification mechanisms to validate voter information against trusted sources such as government databases or biometric records.
* Utilize blockchain's transparent ledger to maintain an auditable record of voter registrations, ensuring accountability and preventing duplicate or fraudulent registrations.

**4. Ballot Casting and Encryption:**

* Develop an intuitive voting interface that enables voters to cast their ballots securely and anonymously.
* Utilize cryptographic techniques such as zero-knowledge proofs or homomorphic encryption to ensure the confidentiality of votes while preserving their integrity.
* Implement mechanisms to prevent double-spending or tampering with ballots, ensuring that each voter can cast only one vote and that votes are recorded accurately.

**5. Vote Tallying and Result Publication**:

* Implement algorithms for aggregating and tallying votes stored on the blockchain, ensuring accuracy and transparency in the tabulation process.
* Enable real-time monitoring of election results, allowing stakeholders to track the progress and outcome of the election as votes are counted.
* Publish election results on the blockchain in a transparent and verifiable manner, providing conclusive evidence of the integrity and fairness of the electoral process.

**6. Security and Auditing:**

* Implement robust security measures to protect the virtual voting system against cyber threats such as hacking, DDoS attacks, and data breaches.
* Conduct regular audits and penetration testing to identify vulnerabilities and ensure compliance with security best practices.
* Enable mechanisms for auditing the integrity of the blockchain network and verifying the accuracy of election results, enhancing trust and confidence in the system.

**7. User Support and Accessibility:**

* Provide comprehensive user support and guidance to facilitate voter participation, including tutorials, FAQs, and troubleshooting resources.
* Ensure accessibility for individuals with disabilities or limited digital literacy through inclusive design principles and assistive technologies.
* Collaborate with relevant stakeholders such as election authorities, advocacy groups, and cybersecurity experts to address user concerns and improve the usability of the virtual voting system.

**1.6 UNIQUE FEATURES OF SYSTEM**

One unique feature of the proposed virtual voting system leveraging blockchain technology is its immutable and transparent audit trail.

Traditionally, the process of auditing election results can be opaque and time-consuming, often relying on manual verification methods that may be prone to errors or manipulation. However, by leveraging blockchain's immutable ledger, the proposed system provides a transparent and tamper-proof record of all voting activities, from voter registration to result publication.

This unique feature offers several advantages:

1. **Transparency:** The blockchain ledger provides real-time visibility into the entire voting process, allowing stakeholders to monitor and audit each transaction securely. This transparency fosters trust in the electoral process by enabling independent verification of election results.
2. **Immutability:** Once recorded on the blockchain, voting data becomes immutable, meaning it cannot be altered or deleted without consensus from the network participants. This ensures the integrity of the electoral data and prevents tampering or manipulation of election results.
3. **Verifiability:** The transparent nature of the blockchain enables voters to verify that their votes have been accurately recorded and included in the final tally. By providing cryptographic proof of participation, voters can independently verify the integrity of the election results, enhancing confidence in the outcome.
4. **Auditability:** Election authorities, auditors, and other stakeholders can conduct comprehensive audits of the voting process using the blockchain's transparent audit trail. This facilitates post-election analysis and investigation of any discrepancies or irregularities, ensuring accountability and fairness in the electoral process.

**CHAPTER 2**

**REQUIREMENT ANALYSIS & SYSTEM SPECIFICATION**

**2.1 FEASIBILITY STUDY**

**1. Technical Feasibility:**

Blockchain Technology: Evaluate the feasibility of implementing blockchain technology, specifically Ethereum, for the virtual voting system. Consider factors such as transaction throughput, smart contract capabilities, and scalability.

Smart Contract Development: Assess the technical feasibility of developing smart contracts to manage the voting process securely. Consider the complexity of the voting logic and the ability to enforce voting rules on the blockchain.

Integration with ReactJS: Determine the feasibility of integrating Ethereum smart contracts with the ReactJS frontend. Ensure compatibility between the technologies and evaluate any challenges in communication and data exchange.

**2. Economic Feasibility**:

Cost Analysis: Conduct a cost analysis to estimate the expenses associated with developing and deploying the virtual voting system. Include costs for software development, blockchain infrastructure, security measures, and ongoing maintenance.

Return on Investment (ROI): Evaluate the potential benefits of the virtual voting system, such as increased transparency, reduced costs of traditional voting methods, and improved voter accessibility. Compare these benefits with the projected costs to determine the ROI of the project.

**3. Operational Feasibility:**

User Acceptance: Assess the willingness of voters to use a virtual voting system based on blockchain technology. Consider conducting surveys or focus groups to gauge user acceptance and identify any concerns or reservations.

Stakeholder Support: Evaluate the support and collaboration from relevant stakeholders, including government agencies, election officials, and regulatory bodies. Ensure alignment with their requirements and regulations for electronic voting systems.

**4. Legal and Regulatory Feasibility:**

Security and Privacy: Evaluate the feasibility of implementing robust security and privacy measures to protect voter data and ensure the integrity of the voting process. Address any legal implications of using blockchain technology for voting.

**5. Schedule Feasibility:**

Project Timeline: Develop a realistic timeline for the development, testing, and deployment of the virtual voting system. Consider factors such as resource availability, technology readiness, and potential setbacks.

Dependencies: Identify any dependencies or external factors that may impact the project schedule. Mitigate risks by addressing potential bottlenecks or delays proactively.

**6. Conclusion:**

Based on the findings of the feasibility study, provide a conclusion regarding the overall feasibility of the virtual voting system project. Highlight any risks or challenges identified and propose recommendations for mitigating them.

By conducting a thorough feasibility study, you can assess the viability of your virtual voting system project and make informed decisions about its development and implementation.

**2.2 SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT**

**1. Introduction**

1.1 Purpose:

The purpose of this document is to provide a comprehensive overview of the software requirements for the development of the virtual voting system using blockchain technology.

1.2 Scope:

The virtual voting system aims to provide a secure, transparent, and accessible platform for voters to cast their votes electronically. The system will utilize blockchain technology, specifically Ethereum, for vote recording and smart contracts for executing voting logic.

**2. Data Requirements**

2.1 Voter Information:

* Name
* Unique Identifier
* Address
* Contact Information
* Authentication Credentials (e.g., Public Key)

2.2 Voting Data:

* Ballot Options
* Timestamp of Vote
* Unique Transaction ID
* Voter Authentication Details

**3. Functional Requirements**

3.1 User Registration:

Users should be able to register on the platform by providing necessary information. The system should verify the authenticity of user information and prevent duplicate registrations.

3.2 Voting Process:

Voters should be able to access the ballot and cast their votes securely. The system should ensure that each voter can cast only one vote per election.Votes should be recorded on the blockchain in a tamper-proof manner.

3.3 Vote Counting:

The system should accurately count the votes and calculate the election results based on the recorded data. Smart contracts should execute the vote counting process transparently and securely.

3.4 Result Announcement:

Election results should be announced publicly once the voting period is complete. The system should provide mechanisms for auditing and verifying the election results.

**4. Performance Requirements**

4.1 Transaction Speed:

The system should handle a large number of transactions efficiently, especially during peak voting times. Transactions should be processed within a reasonable time frame to ensure a smooth voting experience for users.

4.2 Scalability:

The system should be scalable to accommodate a growing number of users and transactions over time. Scalability solutions, such as layer 2 solutions or sidechains, should be considered to enhance performance.

**5. Maintainability Requirements**

5.1 Code Maintainability:

The codebase should be well-structured, modular, and well-documented to facilitate ease of maintenance and future enhancements. Version control systems should be used to track changes and manage code updates effectively.

5.2 System Upgrades:

The system architecture should allow for seamless upgrades and updates to incorporate new features or address security vulnerabilities. Backward compatibility with existing versions should be maintained whenever possible.

**6. Security Requirements**

6.1 Data Security:

Voter information and voting data should be encrypted to protect against unauthorized access or tampering. Access controls should be implemented to restrict data access to authorized personnel only.

6.2 Smart Contract Security:

Smart contracts should be developed following best practices to minimize the risk of vulnerabilities, such as reentrancy attacks or integer overflow. Code audits and formal verification techniques should be employed to ensure the security of smart contracts.

6.3 Authentication and Authorization:

Strong authentication mechanisms should be implemented to verify the identity of voters.

Role-based access controls should be enforced to regulate access to system functionalities based on user roles.

**7. Conclusion**

This Software Requirement Specification document outlines the data, functional, performance, maintainability, and security requirements for the development of the virtual voting system. Adherence to these requirements will ensure the successful implementation of a secure and reliable electronic voting platform using blockchain technology.

**2.3 SDLC MODEL TO BE USED**

For the development of the virtual voting system using blockchain technology, an iterative and flexible software development life cycle (SDLC) model is recommended to accommodate the complexity and evolving nature of the project. The Agile SDLC model is well-suited for this purpose. Here's a breakdown of how Agile can be applied:

**Agile Software Development Life Cycle (SDLC) Model:**

**Planning:**

Define project objectives, scope, and requirements through collaboration with stakeholders.

Create a product backlog consisting of user stories and features prioritized based on value and importance.

Iteration Planning:

Break down the product backlog into smaller, manageable units of work (e.g., sprints or iterations). Select user stories or features to be implemented in each iteration based on their priority and complexity.

Development:

Develop the selected user stories or features in the form of working software. Utilize iterative development practices to incrementally build and refine the system functionality. Conduct regular meetings (e.g., daily stand-ups) to review progress, address challenges, and adjust the development plan as needed.

Testing:

Perform continuous testing throughout the development process to ensure the quality and reliability of the software. Adopt automated testing tools and practices to streamline the testing process and identify issues early. Conduct user acceptance testing (UAT) at the end of each iteration to gather feedback from stakeholders and validate system functionality.

Deployment:

Deploy the developed features to a staging environment for further testing and validation. Utilize continuous integration and continuous deployment (CI/CD) pipelines to automate the deployment process and ensure consistent delivery of updates.

Review and Feedback:

Review the completed features and gather feedback from stakeholders to identify areas for improvement. Use retrospective meetings at the end of each iteration to reflect on the development process, discuss lessons learned, and plan for the next iteration.

Iterative Improvement:

Incorporate feedback and lessons learned from previous iterations into subsequent development cycles. Continuously refine and enhance the virtual voting system based on changing requirements, technological advancements, and stakeholder feedback.

**Benefits of Agile SDLC Model for Virtual Voting System:**

Flexibility:

Allows for adaptation to changing requirements and priorities, essential for a project with evolving regulatory and security considerations.

Transparency:

Promotes collaboration and transparency among team members and stakeholders, fostering a shared understanding of project goals and progress.

Quality Assurance:

Facilitates continuous testing and feedback, leading to early detection and resolution of issues, ensuring a high-quality and reliable system.

Stakeholder Involvement:

Encourages active involvement of stakeholders throughout the development process, resulting in a product that meets their needs and expectations.

Incremental Delivery:

Enables the incremental delivery of functionality, allowing stakeholders to see tangible progress and provide feedback early in the development cycle.

By adopting the Agile SDLC model, the development team can effectively manage the complexities of developing a virtual voting system while ensuring transparency, collaboration, and quality throughout the project lifecycle.

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 DETAIL DESIGN**

**1. System Architecture:**

1.1 Components:

Frontend Application: Developed using ReactJS, provides the user interface for voters to interact with the system.

Backend Services: Manages communication with the blockchain network, handles user authentication, and orchestrates system operations.

Blockchain Network: Utilizes Ethereum blockchain for storing voting data and executing smart contracts.

Smart Contracts: Written in Solidity, define the rules and logic of the voting process and are deployed on the Ethereum network.

1.2 Interaction Flow:

Users interact with the frontend application to register, authenticate, and cast their votes.

Frontend communicates with backend services to access blockchain data and perform operations such as vote recording and verification.

Backend services interact with the Ethereum blockchain to execute smart contracts and retrieve voting data.

**2. Data Design:**

2.1 Data Structures:

Voter Information: Name, Unique Identifier, Address, Contact Information.

Voting Data: Ballot Options, Timestamp of Vote, Transaction ID, Voter Authentication Details.

2.2 Blockchain Data:

Immutable Ledger: Records all voting transactions in a decentralized and tamper-proof manner.

Smart Contract State: Stores voting rules, counts, and results in smart contracts deployed on the Ethereum blockchain.

**3. User Interface Design:**

3.1 User Experience (UX):

Intuitive interface for voters to register, authenticate, and cast their votes securely. Clear instructions and feedback mechanisms to guide users through the voting process.

**4. Integration Design:**

4.1 Frontend-Backend Integration:

Implements secure data exchange protocols to ensure confidentiality and integrity of user data.

4.2 Blockchain Integration:

Utilizes truffle.js libraries to interact with the Ethereum blockchain from backend services. Implements secure communication channels and authentication mechanisms for interacting with blockchain nodes.

**5. Security Design:**

5.1 Data Security:

Encrypts sensitive user information stored in the database and during transmission. Implements access controls and permissions to restrict data access to authorized users only.

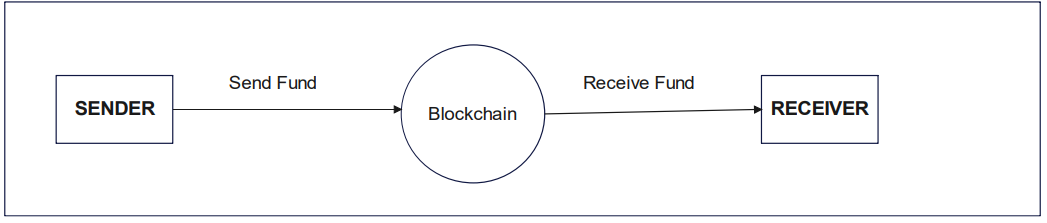
5.2 Smart Contract Security:

Follows best practices for smart contract development to minimize vulnerabilities such as reentrancy attacks and integer overflow. Conducts code audits and formal verification to ensure the security and correctness of smart contract code.

**3.2 SYSTEM DESIGN USING DFD LEVEL 0 AND LEVEL 1**

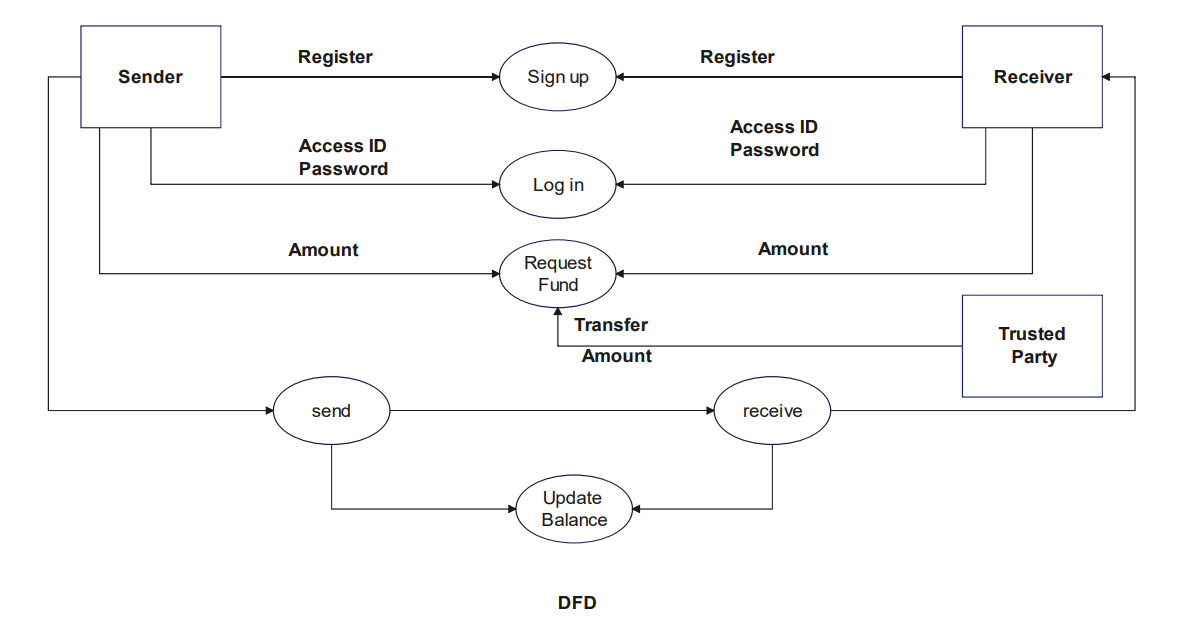
**1. Level 0 DFD:**

At Level 0, the DFD provides an overview of the entire system, showing the interactions between external entities and the system as a whole.



**2. Level 1 DFD :**

At Level 1, the DFD provides a more detailed view of the processes identified in the Level 0 DFD, breaking them down into sub-processes and data flows.



**3.Conclusion:**

The Level 0 and Level 1 DFDs provide a structured representation of the virtual voting system, illustrating the flow of data and interactions between components at different levels of abstraction. These diagrams serve as valuable tools for understanding the system design and can guide the implementation and development process effectively.

**3.3 USE CASE DIAGRAM**

**1.Actors:**

* Voter
* Election Authority

**2. Use Cases:**

2.1 Authenticate User

* Description: Allows voters to verify their identity and authenticate themselves before casting their votes.
* Actors: Voter
* Preconditions: Voters must have registered on the system.
* Postconditions: Voter gains access to the voting interface upon successful authentication.

2.2 Cast Vote

* Description: Enables authenticated users to select their preferred candidates and submit their votes securely.
* Actors: Voter
* Preconditions: Voter must be authenticated.
* Postconditions: Vote is recorded securely in the system.

2.3 Count Votes

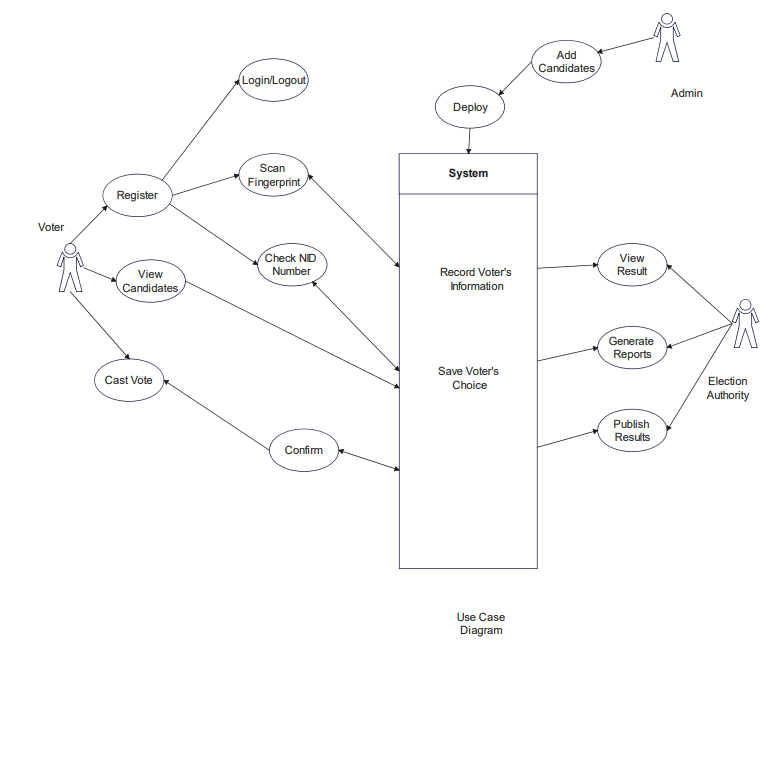
* Description: Calculates the total votes cast for each candidate and determines the election results.
* Actors: Election Authority
* Preconditions: Voting period must have ended.
* Postconditions: Election results are generated and made available to stakeholders.

2.4 Manage Voter Information

* Description: Allows election authorities to manage voter registration, update voter information, and maintain the integrity of voter data.
* Actors: Election Authority
* Preconditions: Access permissions granted to election authorities.
* Postconditions: Voter information is updated and synchronized with the system.

2.5 Announce Election Results

* Description: Informs voters and stakeholders about the outcome of the election by announcing the election results publicly.
* Actors: Election Authority
* Preconditions: Election results must have been calculated.
* Postconditions: Election results are announced and made available to the public.



**3.4 DATABASE DESIGN**

The virtual voting system does not utilize a traditional database due to security concerns, it likely employs a decentralized approach using blockchain technology. In such a system, blockchain serves as the underlying infrastructure for storing and managing voting data securely without the need for a centralized database. Here's an outline of how the system architecture might look without a traditional database:

**1. Blockchain-Based Architecture:**

1.1 Smart Contracts:

* Voter Registry Smart Contract: Stores voter information such as voter ID, authentication credentials, and voting status.
* Ballot Smart Contracts: Define the rules and options for each election ballot, including candidate choices.
* Voting Smart Contracts: Record votes securely on the blockchain, ensuring immutability and transparency.

1.2 Data Storage:

* On-Chain Data: Voter registration details, ballot options, and voting records are stored directly on the blockchain.
* Off-Chain Data: Non-sensitive data or large files (e.g., election results, audit logs) can be stored off-chain in decentralized file storage systems or IPFS (InterPlanetary File System).

**2. Security Considerations:**

2.1 Immutable Ledger:

* Blockchain provides an immutable ledger where all voting transactions are recorded, preventing tampering or manipulation of voting data.
* Transactions are cryptographically secured, ensuring the integrity and authenticity of voting records.

2.2 Decentralization:

* Decentralized architecture eliminates the need for a single point of control, reducing the risk of data breaches or unauthorized access.
* Distributed consensus mechanisms ensure that no single entity can manipulate the voting process or alter the outcome.

2.3 Encryption:

* Voter data and voting transactions can be encrypted to protect privacy and confidentiality.
* Zero-knowledge proofs or homomorphic encryption techniques can be employed to verify votes without revealing sensitive information.

**3. Security Risks Mitigated:**

3.1 Centralized Database Risks:

* Eliminates the risk of centralized databases being compromised or hacked, as there is no central repository of sensitive voter information.
* Protects against insider threats or unauthorized access by distributing data across multiple nodes in the blockchain network.

3.2 Data Tampering:

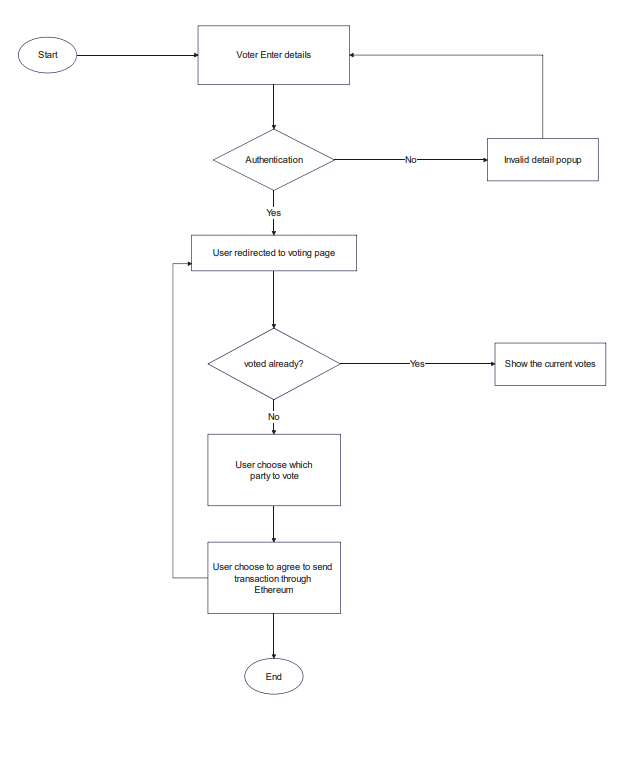
* The immutable nature of blockchain prevents unauthorized modification or tampering of voting data, ensuring the integrity of the electoral process.

3.3 Single Point of Failure:

* Decentralized architecture eliminates the risk of a single point of failure, ensuring the reliability and availability of the voting system.

**3.5**  **DATA FLOW DIAGRAM**

A Data Flow Diagram (DFD) provides a visual representation of the flow of data within a system, illustrating how data moves between processes, data stores, and external entities. For the virtual voting system, a DFD can help in understanding the data flow and interactions between different components of the system.

****

**CHAPTER 4**

**IMPLEMENTATION, TESTING, AND MAINTENANCE**

**4.1 INTRODUCTION TO LANGUAGES, TOOLS AND TECHNOLOGIES USED FOR IMPLEMENTATION**

**1. Blockchain**

Blockchain is a decentralized and distributed digital ledger technology that records transactions across a network of computers. It's the underlying technology behind cryptocurrencies like Bitcoin, but its applications extend far beyond digital currencies. Here's a breakdown of its key components and how it works**:**

* **Decentralization:** Unlike traditional centralized systems where a single authority (like a bank or government) controls the ledger, blockchain operates on a decentralized network of computers (nodes). Each node maintains a copy of the entire blockchain, ensuring that no single entity has control over the data.
* **Blocks**: Transactions are grouped together into blocks. Each block contains a list of transactions, a timestamp, and a reference to the previous block (except for the first block, known as the genesis block).
* **Cryptographic Hashing:** Each block is linked to the previous one through a cryptographic hash function. This creates a chain of blocks, hence the name "blockchain." The hash of a block is a unique identifier generated based on its contents. If someone tries to alter a block, it would change the hash, which would then not match the one stored in the subsequent block, alerting the network to the tampering attempt.
* **Consensus Mechanisms**: Blockchain networks use consensus mechanisms to agree on the validity of transactions and maintain the integrity of the ledger without relying on a central authority. The most common consensus mechanisms include Proof of Work (PoW) and Proof of Stake (PoS), each with its own way of ensuring agreement among network participants.
* **Immutability**: Once a block is added to the blockchain, it is extremely difficult to alter its contents. Because each block references the previous one through its hash, changing the data in one block would require recalculating the hash of all subsequent blocks, as well as having control over the majority of the network's computing power (in the case of PoW) or stake (in the case of PoS), making the blockchain highly secure against tampering.

**2. Smart Contracts**

A smart contract is a self-executing contract with the terms of the agreement directly written into code. It operates on a blockchain platform, such as Ethereum, and automatically executes actions when predefined conditions are met. Smart contracts enable trustless transactions and agreements between parties, eliminating the need for intermediaries and reducing the risk of fraud.

* **Code**: Smart contracts are written in programming languages specifically designed for blockchain platforms, such as Solidity for Ethereum. The code includes the terms and conditions of the contract, as well as the actions to be performed when certain conditions are fulfilled.
* **Deployment:** Once the smart contract code is written, it is deployed onto the blockchain network. This process involves creating a transaction to publish the smart contract code onto the blockchain, making it accessible to all network participants.
* **Execution:** Smart contracts execute automatically when triggered by predefined conditions, often referred to as "if-then" statements. For example, if Party A transfers a certain amount of cryptocurrency to the smart contract, then the smart contract releases the digital asset to Party B. These actions are performed without the need for manual intervention or intermediaries.
* **Immutable**: Once deployed onto the blockchain, smart contracts are immutable, meaning their code cannot be altered or tampered with. This ensures the integrity and security of the contract, as parties can trust that the terms will be executed exactly as specified.
* **Decentralization**: Smart contracts run on decentralized blockchain networks, which means they are not controlled by any single entity. This decentralization ensures transparency, security, and censorship resistance, as no central authority has the power to modify or interfere with the execution of the contract.

**3. Solidity programming language**

Solidity is a high-level programming language specifically designed for writing smart contracts on blockchain platforms, primarily Ethereum. It is statically typed and supports inheritance, libraries, and complex user-defined types among other features. Solidity is essential for building decentralized applications (DApps) and executing smart contracts on the Ethereum Virtual Machine (EVM).

* **Syntax and Structure**: Solidity syntax is similar to that of JavaScript and other C-like languages, making it relatively easy to learn for developers familiar with these languages. Solidity code is organized into contracts, which are similar to classes in object-oriented programming. Each contract contains state variables, functions, and modifiers that define the behavior and functionality of the smart contract.
* **Data Types:** Solidity supports various data types, including integers, booleans, strings, arrays, and structs. It also includes special data types such as address (for Ethereum addresses), bytes (for raw binary data), and mappings (key-value pairs).
* **Functions and Modifiers:** Solidity allows developers to define functions within contracts to encapsulate logic and define behavior. Functions can be called internally or externally, and they can have modifiers to enforce access control, validate inputs, or perform other checks before executing the function's code.
* **Event Logging:** Solidity supports event logging, which allows smart contracts to emit events during execution. These events can be captured by external applications or other smart contracts and used for logging, tracking, or triggering additional actions.
* **Inheritance and Libraries:** Solidity supports inheritance, allowing developers to create hierarchical relationships between contracts and reuse code efficiently. It also supports libraries, which are reusable pieces of code that can be imported into contracts to provide common functionality.
* **Security Considerations:** Writing secure Solidity code is critical, as vulnerabilities in smart contracts can lead to exploits and financial losses. Common security considerations include avoiding reentrancy, carefully managing access control, validating inputs, and using safe arithmetic operations to prevent overflows and underflows.
* **Development Tools**: Several development tools and frameworks are available to assist developers in writing, testing, and deploying Solidity smart contracts. These include the Solidity compiler, development environments like Remix and Truffle, testing frameworks like Mocha and Chai, and Ethereum client implementations like Ganache.

**4.2 TESTING TECHNIQUES AND TEST CASES USED**

# 1. Introduction

In an era characterized by unprecedented technological advancements, the traditional methods of conducting elections have come under increasing scrutiny. The digital age has ushered in the potential for innovation in the electoral process, promising greater transparency, security, and accessibility. One groundbreaking development that has garnered significant attention is the integration of blockchain technology into the realm of voting, giving rise to what is commonly referred to as "virtual voting." Blockchain, originally conceived as the underlying technology for cryptocurrencies like Bitcoin, has demonstrated its ability to revolutionize various industries, and elections are no exception.

The concept of virtual voting using blockchain technology is both revolutionary and disruptive. It offers the promise of a democratic system that is more secure, transparent, and efficient. This novel approach combines the security features of blockchain, such as decentralization and cryptographic protection, with the accessibility and convenience of a virtual voting platform. As a result, it has the potential to address longstanding concerns related to electoral integrity, voter fraud, and accessibility.

## 1.1 Scope

### 1.1.1 In Scope

Scope defines the features, functional or non-functional requirements of the software that will betested. Features of the Project:

1. Candidates registration: the number of candidates registered for the election should be between 2 to 9 less than that or greater than that is not allowed.

2. Address mapping: Every registered voter, candidates and election commission have their address mapped corresponding to every action taken by them to check the security and other aspects required for virtual voting machines.

3. User interface: The virtual voting machine would need to provide a user-friendly interface for users and candidates to register and casting of vote should be smooth.

4. Security: The machine uses blockchain and smart contracts which is highly secure and cannot be tampered in any circumstances.

### 1.1.2 Out of Scope

Out Of Scope defines the features, functional or non-functional requirements of the software that will NOT be tested :

1. Scalability: Load testing to ensure the platform can handle increased user loads

## 1.2 Quality Objective

Here make a mention of the overall objective that you plan to achieve without your testing Some objectives of your testing project could be

Ensure the Application Under Test conforms to functional and nonfunctional requirements. Ensure the AUT meets the quality specifications defined by the client.

Bugs/issues are identified and fixed before going live.

## 1.3 Roles and Responsibilities

Detail description of the Roles and responsibilities of different team members like:

• QA Analyst : Surya Pratap singh

• Test Manager : Prof. Shreela Pareek

• Configuration Manager: Prof. Neha Shukla

• Developers : Utkarsh Mishra, Surya Pratap Singh, Adrika Tripathi

• Installation Team :Prof. Shreela Pareek, Prof. Neha Shukla,Utkarsh Mishra, Surya Pratap Singh, Adrika Tripathi

# 2 Test Methodology

## 2.1 Overview

We are using an iterative testing approach to make sure our project works well. This means we test it in small steps, starting with checking if each part works on its own. Then, we see how different parts work together.

We keep testing as we make changes and add new things. This way, we make sure our project is always working well, even after modification.

## 2.2 Test Levels

**Test Levels define the Types of Testing to be executed on the Application Under Test (AUT**). We aim to test our project at the following levels :

1) Unit Testing: This is the lowest level of testing and focuses on individual components or functions within the software. Developers often perform unit tests to verify that specific parts of the code work correctly.

2) Integration Testing: This level of testing checks how different components or modules of the software work together. It ensures that integrated parts of the software function as intended.

3) System Testing: At this level, the entire system is tested as a whole. It verifies

that the software meets its specified requirements and functions properly in its

intended environment.

## 2.3 Test Completeness

Here you define the criterias that will deem your testing complete. For instance, a few criteria to check Test Completeness would be

• 100% test coverage

• All Manual & Automated Test cases executed

• All open bugs are fixed or will be fixed in next release

**3** **Test Deliverables**

Here are the deliverables:

• Test Plan

• Test Cases

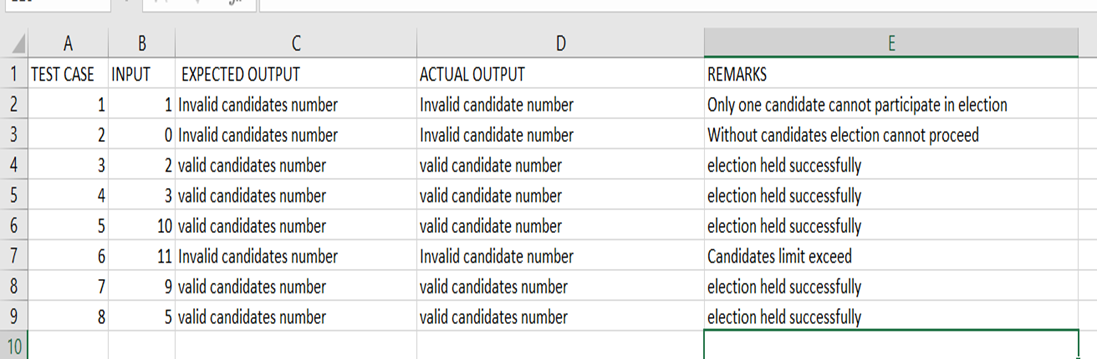
• Bug Reports

• Test Strategy

# 4 Test Cases :

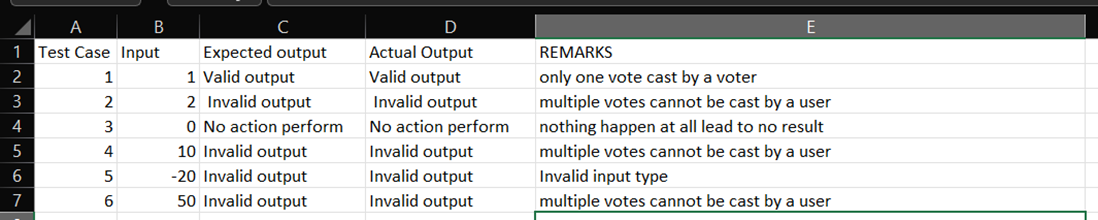
**Boundary Value analysis: Interface Capability Number of candidates = 2 - 10 :**

| **Invalid(min-1)** | **Valid**  **(min, min+1, nominal, max-1, max)** | **Invalid (max+1)** |
| --- | --- | --- |
| **0,1** | **2, 3, 5, 9, 10** | **11** |

****

**Equivalence Class Testing: Interface Capability No of votes acceptable by a voter = 1**

| **Invalid** | **valid** | **Invalid** |
| --- | --- | --- |
| **0** | **1** | **2,3,4…………** |

****

**Equivalence Class testing: Input File Verification and Classification**

| Accepted address | Not Accepted address |
| --- | --- |
| 20 byte address | other than 20 byte address |

**Decision Table: Input File Verification, File Classification & Interface capability:**

| **Conditions** | **Input-1** | **Input-2** | **Input-3** | **Input-4** |
| --- | --- | --- | --- | --- |
| **address size** | **T** | **T** | **F** | **F** |
| **registered person (VOTER)** | **T** | **F** | **T** | **F** |
| **Result** | **Accepted** | **Not Accepted** | **Not Accepted** | **Not Accepted** |

**Example:**

| **Conditions** | **Input-1** | **Input-2** | **Input-3** | **Input-4** | **Input-5** | **Input-6** |
| --- | --- | --- | --- | --- | --- | --- |
| **address size** | **20 byte** | **20 byte** | **20 byte** | **>20 byte** | **<20 byte** | **<20 byte** |
| **registered person (Voter)** | **same address** | **Election Commission address** | **Candidate address** | **same address** | **Election Commission address** | **Candidate address** |
| **Result** | **Accepted** | **Not**  **Accepted** | **NOT**  **Accepted** | **Not Accepted** | **NOT**  **Accepted** | **Not Accepted** |

**5.**  **Resource & Environment Needs**

## 5.1 Testing Tools

List of tools like:

1. Selenium
2. Mentis BT
3. Automation BT

**5.2 Test Environment**

It mentions the minimum **hardware** requirements that will be used to test the Application. Following **software is** required in addition to client-specific software.

• Windows 10 and above preferred

• VSCode 2022 or above preferred

• Chrome, Mozilla or Edge Preferred over non-chromium based browsers

**CHAPTER 5**

**RESULTS AND DISCUSSIONS**

**5.1 USER INTERFACE REPRESENTATION**

Representing the user interface (UI) of the virtual voting system is crucial for ensuring an intuitive and user-friendly voting experience. Here's how the UI can be represented:

**1. Mockups:**

Mockups are static representations of the UI design, showcasing the layout, structure, and visual elements of the voting system. They provide stakeholders with a clear visual understanding of the system's UI without the need for functional implementation. Mockups can be created using design tools such as Sketch, Adobe XD, or Figma.

**2. Wireframes:**

Wireframes are skeletal outlines of the UI design, focusing on the arrangement of elements and functionality rather than visual aesthetics. They serve as a blueprint for the UI layout and navigation flow. Wireframes can be created using wireframing tools like Balsamiq, Axure, or Adobe Illustrator.

**3. Prototypes:**

Prototypes are interactive representations of the UI design, allowing stakeholders to experience the voting system's functionality and navigation flow. They provide a realistic simulation of how users will interact with the system. Prototypes can be created using prototyping tools such as InVision, Proto.io, or Adobe XD.

**4. UI Components:**

4.1. Voter Registration:

* Input fields for entering personal information (name, address, contact details).
* Authentication credentials (username, password).
* Buttons for submitting registration details and creating an account.

4.2. Voting Interface:

* Ballot options displayed in a list or grid format.
* Instructions and prompts guiding users through the voting process.

4.3. Election Results:

* Tabular of election results.
* Filters and sorting options for viewing results by category or candidate.
* Visual indicators for displaying vote counts.

**5. Accessibility Considerations:**

* Ensure that the UI design complies with accessibility standards, providing options for users with disabilities to navigate and interact with the system effectively.
* Use proper contrast ratios, font sizes, and resizable elements to accommodate users with visual impairments.
* Provide alternative text for images and multimedia content to facilitate screen reader accessibility.

**5.2 BRIEF DESCRIPTION OF VARIOUS MODULES OF THE SYSTEM**

**1. Voter Registration Module**:

* User Registration: Allows voters to register on the platform by providing necessary personal information.
* Authentication: Verifies the identity of voters before granting access to the voting interface.
* Profile Management: Enables voters to update their profile information, such as contact details or password.

**2. Candidate Module:**

* Candidate Registration: Allows candidates to register for elections by providing their personal information and candidacy details.
* Candidate Profile: Displays information about registered candidates, including their platform, bio, and campaign details.
* Candidate Management: Enables election authorities to manage candidate registrations and verify candidate eligibility.

**3. Election Commission Module:**

* Election Setup: Facilitates the setup and configuration of election parameters, including date, time, and ballot options.
* Ballot Management: Allows election authorities to create, modify, and monitor election ballots.
* Result Declaration: Provides tools for election authorities to tabulate votes and declare election results securely.

**4. Blockchain Transaction Module**:

* Ganache Integration: Integrates with Ganache, a local Ethereum blockchain for testing and development purposes.
* Transaction Recording: Records voting transactions on the blockchain using smart contracts.
* Transaction Verification: Verifies the integrity and authenticity of voting transactions using blockchain technology.

**5.3 SNAPSHOTS OF SYSTEM WITH BRIEF DETAIL OF EACH**

**1.Home Page:**

1.1Connect MetaMask:

* Provide a prominent button or link labeled "Connect MetaMask" on the home page.
* When clicked, the system prompts the user to connect their MetaMask wallet.
* Display a message indicating that MetaMask connection is required for voting.

1.2 Select Account:

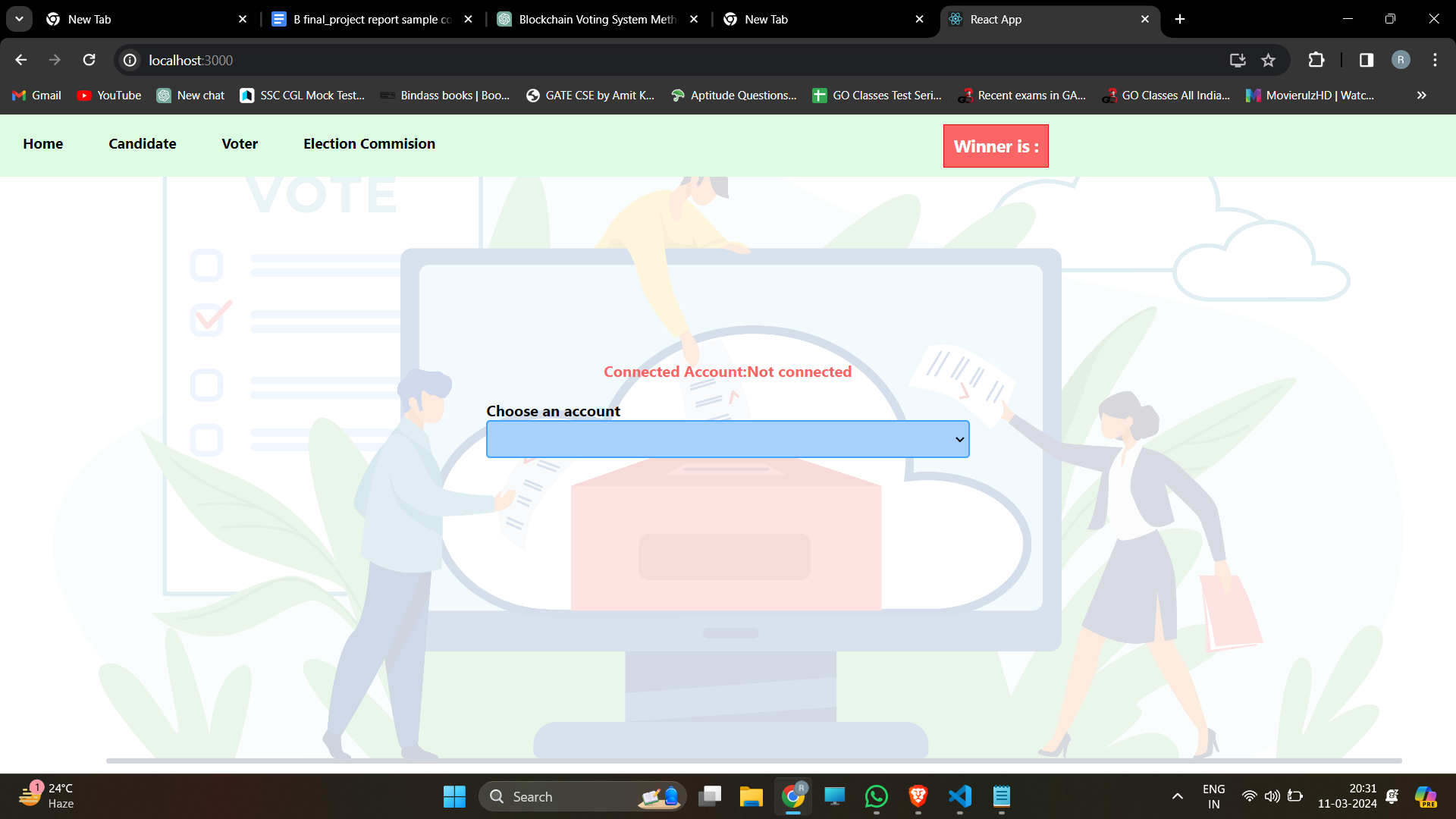
* Upon successful MetaMask connection, display a list of available accounts from Ganache.
* Each account should be represented with its address and balance.
* Allow users to select their desired account by clicking on it.

1.3 Confirmation:

* Once the user selects an account, display a confirmation message asking the user to confirm their selection.
* Provide options to confirm or cancel the account selection.

1.4Proceed to Voting:

* After confirming the account selection, provide a button or link to proceed to the voting interface.
* Ensure that the selected account is securely associated with the user's voting session.



**2.Single Account Usage:**

2.1 Account Selection:

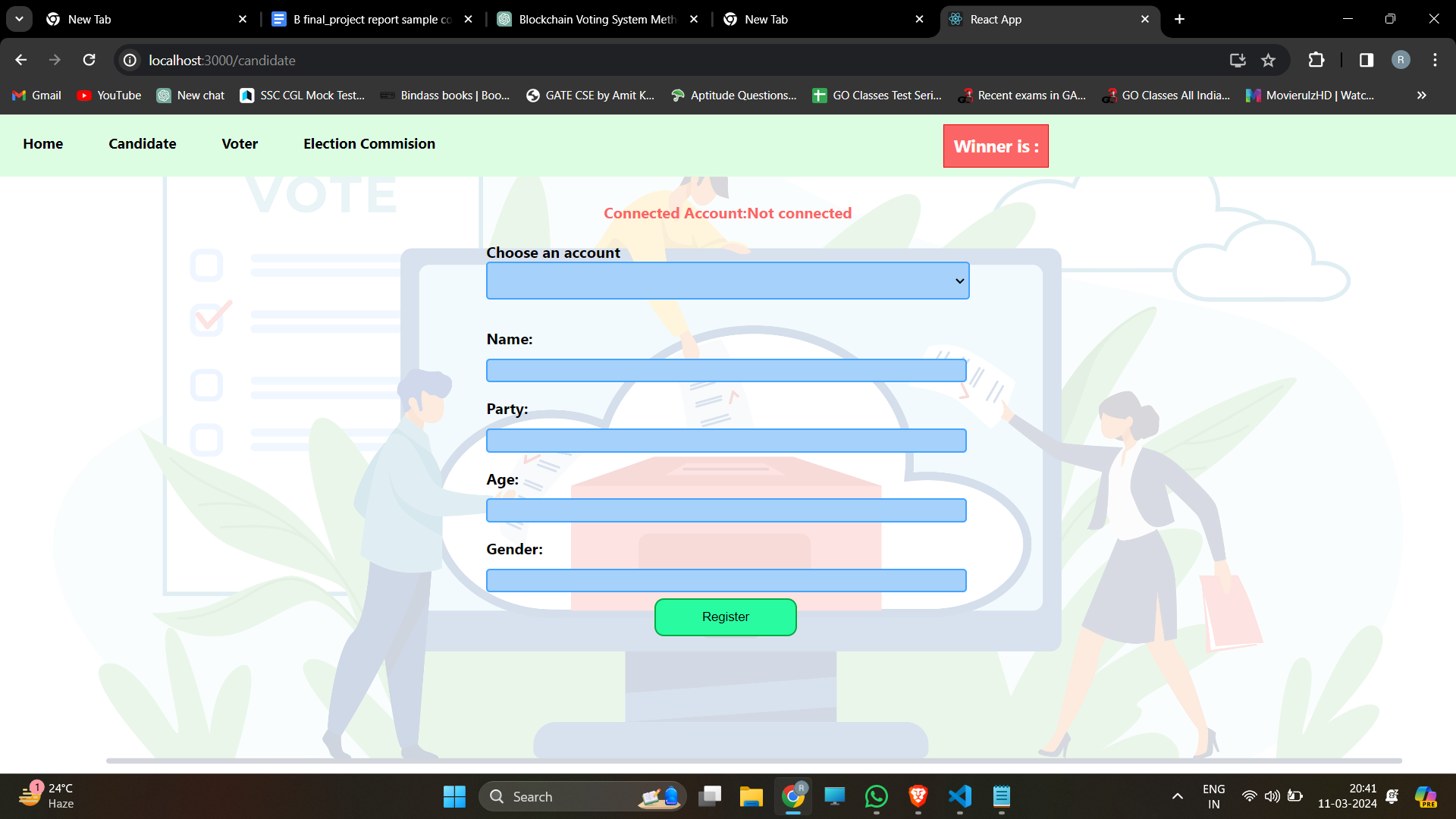
* On the home page or during the registration process, users are prompted to select a Ganache account to use for voting.
* The system provides a dropdown or list of available Ganache accounts, allowing users to choose one.

2.2 Registration:

* Once an account is selected, users proceed with the registration process as usual.
* The user's registration information is associated with the selected Ganache account.

2.3 Voting:

* When it's time to vote, users log in with their selected Ganache account.
* The system verifies the account and allows the user to cast their vote.



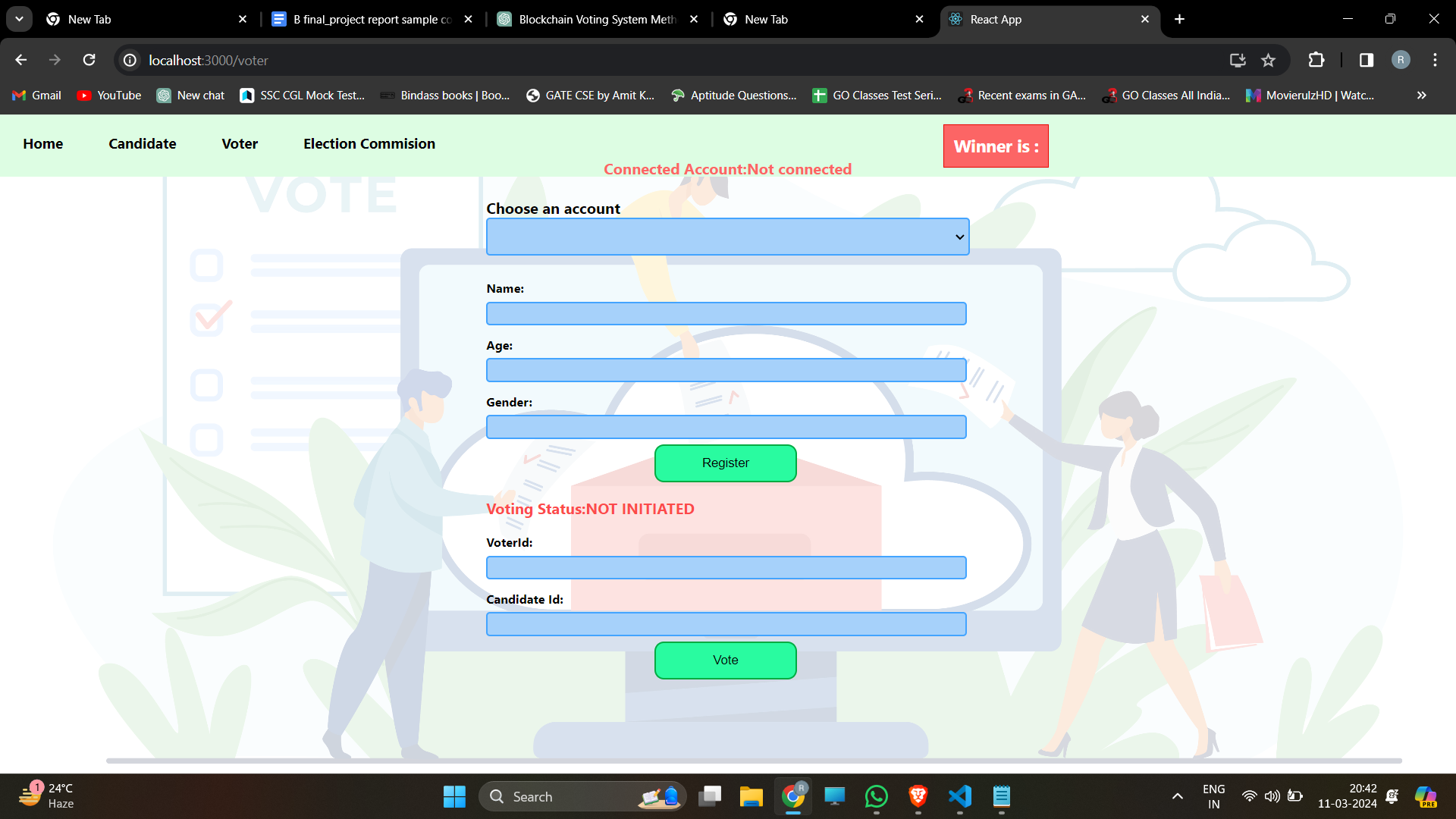
**3. Implementation Considerations:**

3.1Account Verification:

* Implement mechanisms to verify the authenticity of the selected Ganache account during registration and voting.
* Use cryptographic techniques or unique identifiers to ensure that each account is associated with a legitimate voter.

3.2User Identification:

* While users may share a Ganache account, ensure that each user is uniquely identified within the system.
* Use additional user information or authentication methods to differentiate between users sharing the same account.



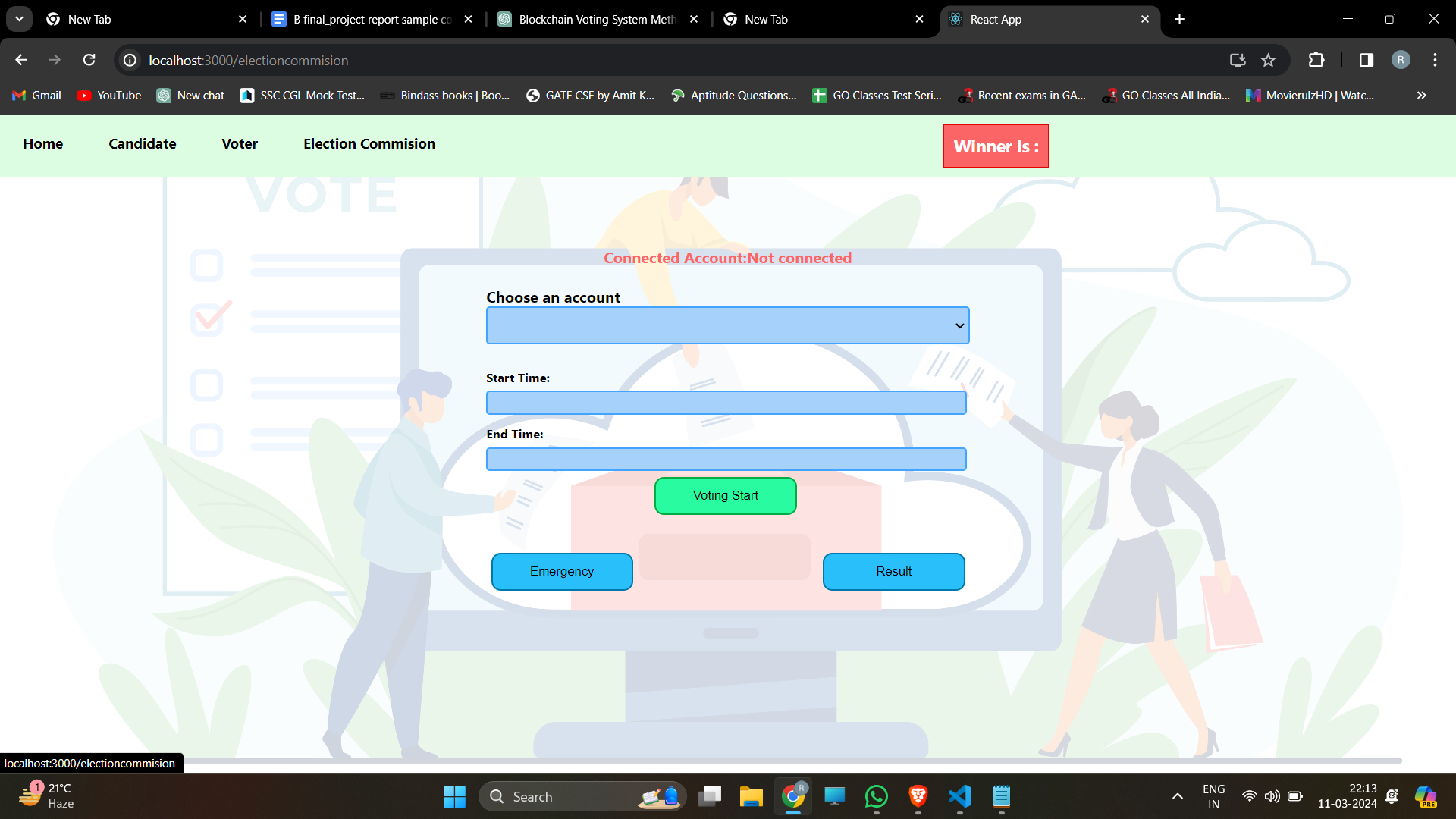
3.3Transaction Signing:

* Utilize MetaMask or similar tools to enable users to sign transactions with their Ganache account securely.
* Ensure that each transaction is properly signed and authenticated to prevent unauthorized voting.

**4.The Election Commission** module serves as the administrative backbone of the virtual voting system, responsible for managing and overseeing all aspects of the electoral process. Here's an overview of the functionalities and responsibilities of the Election Commission module:

4.1 Election Setup:

* Configuration: Allows election authorities to configure election parameters, including date, time, and ballot options.
* Validation: Ensures that all election settings comply with legal requirements and organizational policies.



* Preview: Provides a preview of the election setup for review and approval before activation.

4.2 Ballot Management:

* Creation: Facilitates the creation of election ballots, including candidate options and voting rules.
* Modification: Allows election authorities to modify ballot contents or parameters if necessary.
* Monitoring: Provides real-time monitoring of ballots, including status, participation metrics, and voter turnout.

4.3 Candidate Management:

* Registration: Enables candidates to register for elections by providing their personal information and candidacy details.
* Verification: Verifies candidate eligibility and ensures that all registration requirements are met.
* Campaign Oversight: Monitors and regulates candidate campaign activities to ensure compliance with election regulations.

4.4 Voter Registration:

* Validation: Validates voter registrations to verify eligibility and prevent fraudulent registrations.
* Approval: Approves registered voters for participation in the election after verification.
* Rejection: Rejects registrations that do not meet the required criteria or fail validation checks.

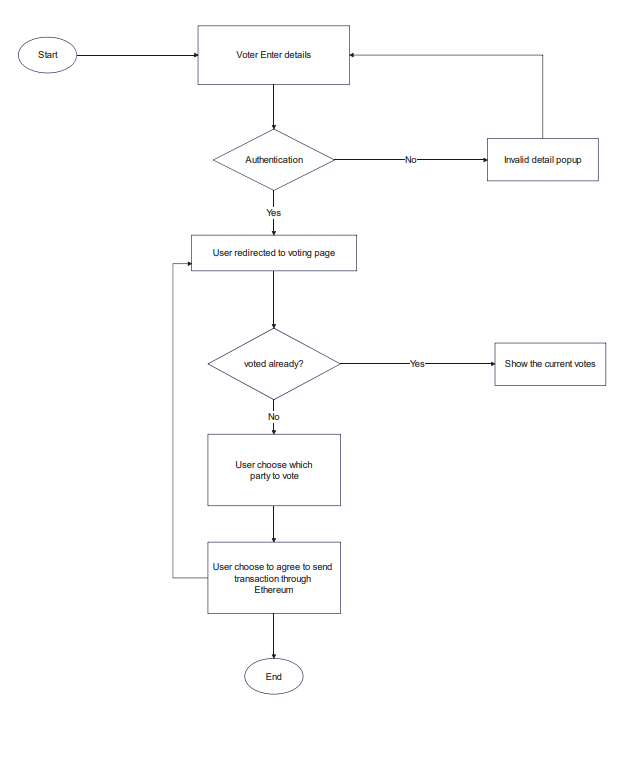
4.5 Result Declaration:

* Vote Tabulation: Aggregates and tallies the votes cast for each candidate or ballot option.
* Result Calculation: Calculates election results based on the vote counts and predefined voting rules.
* Announcement: Officially announces election results to the public and stakeholders in a transparent and timely manner.

**5.4 BACK-ENDS REPRESENTATION**

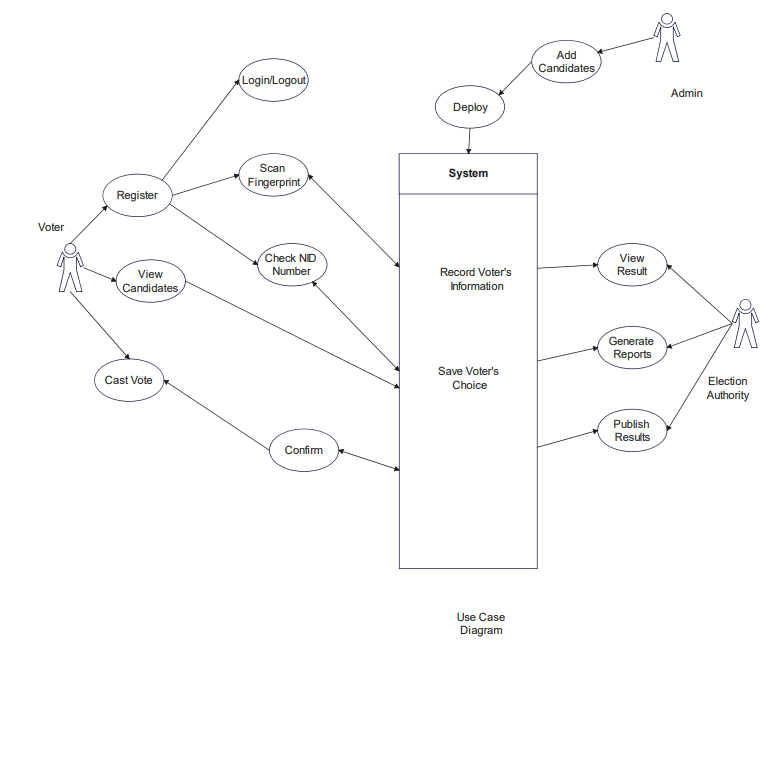
**1. Flowcharts:**

* Process Flow: Visualize the flow of data and control within the backend application for different voting system workflows, such as voter registration, candidate registration, ballot creation, vote casting, and result declaration.
* Decision Trees: Represent decision-making processes within the backend logic, such as user authentication, eligibility verification, data validation, and error handling.
* Error Handling: Diagram how the backend detects, handles, and responds to errors and exceptions during the voting process to ensure the integrity and security of the system.

****

**2. UML Diagrams:**

* Class Diagram: Show the classes and their relationships within the backend application, including entities like users, candidates, ballots, and votes, as well as controllers, services, and utilities.
* Sequence Diagram: Demonstrate the sequence of interactions between backend components and external entities for various voting system processes, such as voter registration, ballot creation, vote casting, and result calculation.
* Component Diagram: Provide an overview of the structural organization of the backend, highlighting the major components and their dependencies, such as the integration with external services like MetaMask for blockchain interaction.detects, handles, and responds to errors and exceptions during the voting process to ensure the integrity and security of the system.



**CHAPTER 6**

**CONCLUSION AND FUTURE SCOPE**

**6.1 CONCLUSION AND FUTURE SCOPE**

Hundreds of transactions per second might be sent onto using every component of the smart contract and the blockchain. Because of the block chain's openness, elections may be more thoroughly audited and understood. These characteristics are some of what a voting system must have. These decentralized networks provide properties that can bring forth more democratic electoral procedures, especially to control electoral processes. To make e-voting more popular a possible open, transparent, and independently auditable Its foundation should be blockchain technology. This initiative investigates blockchain's possibilities technology's role in the electronic voting system. The Blockchain will be widely dispersed and publicly verified such that it cannot be tampered with by anyone.

To reduce the blockchain's workload for nations with a larger population, further precautions would be required to enable higher transaction throughput per second. Making the system more secure. Integrating biometric authentication i.e fingerprint ,an iris scan and retina scan for better authentication.It is essential to continually assess and improve the security features of blockchain systems to stay ahead of emerging threats and ensure the continued trust of the electorate.

**REFERENCES**

1. Linh Vo-Cao-Thuy, Khoi Cao-Minh, Chuong Dang-Le-Bao and Tuan A. Nguyen,(2019), Votereum: An Ethereum-based E-voting system :
2. Vaibhav Anasune, Pradeep Choudhari , Madhura Kelapure and Pranali Shirke Prasad Halgaonkar,“Online Voting: Voting System Using B-chain”,(2019), Online Voting: Voting System Using Blockchain
3. David Khoury,Elie F. Kfoury, Ali Kassem and Hamza Harb,(2018), Decentralized Voting Platform Based on Ethereum Blockchain
4. G Bhavan,i ,(2018) ,“Survey on Blockchain Based E-Voting Recording System Design”
5. Friðrik Þ. Hjálmarsson , Gunnlaugur K . Hreiðarsson, ,(2018)“Blockchain-Based E-Voting System”
6. Design:Rifa Hanifatunnisa and Budi Rahardjo,(2017) , Blockchain Based E-Voting Recording System
7. R. L. Rivest, A. Shamir, and Y. Tauman, “How to Leak a Secret: Theory and Applications of Ring Signatures,” in Theoretical Computer Science - Essays in Memory of Shimon Even, 2006
8. Hsiao JH, Tso R, Chen CM, Wu ME. ‘Decentralized EVoting Systems Based on Blockchain Technology’, Advances in Computer Science and Ubiquitous Computing. CUTE 2017, CSA 2017. Lecture Notes in Electrical Engineering, vol 474. Springer, Singapore; c2018.
9. Emre Yavuz ; Ali Kaan Koç ; Umut Can Çabuk ; Gökhan Dalkılıç (2018) Towards secure e-voting using ethereum blockchain .
10. Y. Takabatake, D. Kotani, and Y. Okabe, “An anonymous distributed electronic voting system using Zerocoin
11. Venkata Naga Rani B, Akshay S, Arun kumar M , Ishwar Kumar M A , (2019) , Decentralized E-Voting System,International Research Journal of Engineering and Technology
12. Hoyul Choi, Hyunsoo Kwon, Junbeom Hur, (2015) ,“A secure OTP algorithm using a smartphone application”, Seventh International Conference on Ubiquitous and Future Networks, Sapporo, Japan.
13. G. Wood,[2014] "Ethereum: a secure decentralized generalized transaction ledger", Ethereum Project Yellow Paper,
14. E. Maaten,[2004] “Towards remote e-voting: Estonian case”, Electronic Voting in Europe-Technology, Law, Politics and Society,.
15. Estonian National Electoral Committee “E-voting System”, 2010.