**Blockchain-Based Voting System**

**A Project Report**

Submitted in partial fulfilment of the Requirements for the award of the Degree of

##### BACHELOR OF SCIENCE (INFORMATIONTECHNOLOGY)

By

Parth Raja KCTYBSCIT037

& Ansh Kothari

KCTYBSCIT024

**Under the esteemed guidance of Head of Department**

**Dr. Rakhi Gupta**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

##### KISHINCHAND CHELLARAM COLLEGE

***(Affiliated to HSNC University)* Mumbai, 400020 MAHARASHTRA**

**2024-25**

#### PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PRN No.: 2022230210020051 Roll No: KCTYBSCIT037

2022230210020009 KCTYBSCIT024

**Name of the Student: Parth Raja**

**Title of the Project: Blockchain-Based Voting System**

**Name of the Guide: Dr Rakhi Gupta Teaching experience of the Guide: 29 Years**

Is this your first submission? Yes  No 

Signature of the Student Signature of the Guide

Date: ………………… Date: …………………….

Signature of the Co-Ordinator

Date: …………………

##### KISHINCHAND CHELLARAM COLLEGE

**(*Affiliated to HSNC University, Mumbai)***

##### MUMBAI, MAHARASHTRA - 400020 DEPARTMENT OF INFORMATION TECHNOLOGY

****

**CERTIFICATE**

This is to certify that the project entitled, "**Blockchain-Based Voting System**", is Bonafide work of Parth Raja bearing KCTYBSCIT037 and Ansh Kothari bearing KCTYBSCIT024 submitted in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from

HSNCU University.

**Internal Guide Co-Ordinator**

**External Examiner**

**Date: College Seal**

## ACKNOWLEDGEMENT

We would like to take this opportunity to express a deep sense of gratitude to Dr. Tejashree V. Shanbhag (Principal of K.C. College) and Dr. Rakhi Gupta (H.O.D. of B.Sc. I.T. Department) for their cordial support, valuable information and guidance, which helped us in completing this task through various stages. We take this opportunity to express our profound gratitude and deep regards to our guide, Dr Rakhi Gupta for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her time-to-time shall care us a long way in the journey of life on which we are about to embark.

We are obliged to staff members of K.C. College for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of our project.

Ansh Kothari

Parth Raja

## DECLARATION

I hereby declare that the project entitled, **“Blockchain-Based Voting System”** done at **K.C. College** is done, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

**Name and Signature of the Student**

## TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page No** |
| 1 | Introduction | 10 |
|  | 1.1 Background | 11 |
| 1.2 Objectives | 12 |
| 1.3 Purpose, Scope and Applicability | 12 |
| 1.4 Achievements | 13 |
| 1.5 Organization of Report | 14 |
| 2 | Survey of Technology | 16 |
|  | 2.1 Front-End Features | 17 |
| 2.2 Back-End Features | 18 |
| 2.3 Justification | 19 |
| 3 | Requirement and Analysis | 20 |
|  | 3.1 Problem Definition | 21 |
| 3.2 Requirement Specification | 21 |
| 3.3 Software and Hardware Requirement | 22 |
| 3.4 Planning and Scheduling | 23 |
| 3.5 Gantt Chart | 24 |
| 4 | System Design | 25 |
|  | 4.1 Data Flow Diagram | 26 |
| 4.2 Class Diagram | 28 |
| 4.3 Use Case Diagram | 30 |
| 4.4 Activity Diagram | 32 |
| 5 | Implementing and Testing | 34 |
|  | * 1. Implementation Approaches   2. Code Details and Efficiency   3. Testing Approaches   4. Modification and Improvement   5. Test Cases | 35  35  43  45  46 |

|  |  |  |
| --- | --- | --- |
| 6 | Results and Discussion | 47 |
|  | * 1. Test Reports   2. Results | 48  55 |
| 7 | Conclusion | 59 |
|  | * 1. Conclusion   2. Future Scope of the Project | 60  60 |
| 8 | Cost and Benefit Analysis | 61 |
| 9 | User Manual | 64 |
| 10 | References | 68 |
| 11 | Bibliography | 69 |

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **List of Tables** | **Page No.** |
| 1. | Table 4.1 Data Flow Diagram Symbols | 26 |
| 2. | Table 4.2 Class Diagram Symbols | 26 |
| 3. | Table 4.3 Use Case Diagram Symbols | 28 |
| 4. | Table 4.4 Activity Diagram Symbols | 30 |

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **List of Figures** | **Page No.** |
| 1. | Fig 3.1 Gantt Chart | 24 |
| 2. | Fig 4.1 Data Flow Diagram Level 0 | 23 |
| 3. | Fig 4.2 Data Flow Diagram Level 1 | 24 |
| 4. | Fig 4.3 Class Diagram | 27 |
| 5. | Fig 4.4 Use Case Diagram | 29 |
| 6. | Fig 4.5 Activity Diagram | 31 |

|  |  |  |
| --- | --- | --- |
| 7. | Fig 6.1-6.3 Add Candidate | 48 |
| 8. | Fig 6.4-6.5 Edit Candidate | 49 |
| 9. | Fig 6.6 Delete Candidate | 50 |
| 10. | Fig 6.7-6.8 Register Voter | 50 |
| 11. | Fig 6.9 Voter Aadhar Verification | 51 |
| 12. | Fig 6.11 Caste Vote | 52 |
| 13. | Fig 6.12 Double Voting | 53 |
| 14. | Fig 6.14 Duplicate Voter Registration | 53 |
| 15. | Fig 6.15 Start Voting | 54 |
| 16. | Fig 6.16 Stop Voting | 54 |
| 17. | Fig 6.17 Home Page | 55 |
| 18. | Fig 6.18 Voter Registration | 56 |
| 19. | Fig 6.19 & 6.20 Admin Panel | 56 |
| 20. | Fig 6.21 Viewing results | 57 |
| 21. | Fig 6.22 Voter’s Dashboard | 57 |
| 22. | Fig 6.23 Register Voter (Aadhar XML Required) | 58 |
| 23. | Fig 6.24 Voting Process guide(English/Hindi) | 58 |

**ABSTRACT**

The advent of blockchain technology presents significant opportunities to enhance the security, transparency, and integrity of voting systems. This project explores the implementation of a blockchain-based voting system designed to address common issues associated with traditional voting methods, such as fraud, tampering, and lack of transparency. Leveraging the decentralized and immutable nature of blockchain, the proposed system aims to ensure that each vote is accurately recorded and securely stored, with results that are verifiable by all stakeholders. The system employs a combination of cryptographic techniques and smart contracts to facilitate secure voter registration, anonymous ballot casting, and transparent vote tallying. By utilizing a permissioned blockchain, we ensure that only authorized entities can participate in the consensus process, maintaining the balance between security and performance.

This project involves the development of a prototype application using tools such as Ethereum, with a focus on user experience, scalability, and real-world applicability. Preliminary results indicate that blockchain can significantly enhance the trustworthiness and efficiency of electoral processes, paving the way for more robust democratic practices.

Keywords: Blockchain, Voting System, Security, Transparency, Cryptography, Smart Contracts, Permissioned Blockchain, Ethereum, Electoral Processes.

# CHAPTER 1

## INTRODUCTION

##### BACKGROUND

* The bedrock of any democratic society lies in the integrity and transparency of its electoral processes. Ensuring the accurate representation of the popular will is paramount, yet traditional voting systems, encompassing both paper- based and electronic methods, remain susceptible to vulnerabilities such as fraudulent activities, manipulation, and a lack of transparency. These concerns erode public trust and necessitate the exploration of innovative solutions capable of fortifying the electoral infrastructure.
* Recent advancements in blockchain technology have presented a compelling opportunity to address these longstanding challenges. Blockchain, with its inherent characteristics of decentralization and immutability, offers a novel approach to enhance the security and trustworthiness of voting systems. By leveraging a distributed ledger, every transaction, in this case, every vote cast, is cryptographically secured and permanently recorded, making it virtually tamper-proof. This inherent auditability fosters transparency and allows for independent verification of election results, bolstering public confidence in the democratic process.
* This project delves into the development and implementation of a blockchain-based voting system, aiming to showcase its potential to revolutionize elections. Through a comprehensive analysis, we will investigate the multifaceted benefits of this technology, including enhanced security, increased transparency, and improved accessibility. Furthermore, the project will also address the inherent challenges associated with blockchain implementation, such as scalability, usability, and the need for robust regulatory frameworks. By exploring both the potential and limitations of this innovative approach, this project seeks to contribute to the ongoing discourse surrounding the future of secure and trustworthy elections. The findings will provide valuable insights for stakeholders, policymakers, and technology experts engaged in shaping the evolution of democratic processes in the digital age.

##### OBJECTIVES

* Develop a secure, transparent, and tamper-proof blockchain-based voting system.
* Ensure voter anonymity and protect voter identity for anonymous vote casting.
* Implement privacy-preserving cryptographic techniques to secure voter data and maintain privacy throughout the voting process.
* Utilize robust authentication mechanisms to ensure unique voter identification and prevent duplicate voting.
* Guarantee transparency and enable all stakeholders (voters, candidates, election officials) to verify the integrity of the election results.
* Leverage the immutability of the blockchain to ensure that election results cannot be altered or manipulated.

###### Purpose, Scope, and Applicability

* + 1. **Purpose**

The purpose of this project is to design and implement a voting system that leverages the unique features of blockchain technology to improve the security, transparency, and trustworthiness of the voting process. By addressing common issues associated with traditional voting methods, the project aims to provide a robust alternative that can be adopted in various electoral scenarios. Additionally, the system integrates Aadhaar Offline Verification (UIDAI-Signed XML File) for voter authentication, ensuring that only legitimate voters participate while preserving privacy by not storing voter’s information and ensuring age and name verification.

###### Scope

The project scope encompasses designing a basic system architecture that leverages blockchain for secure vote recording and counting. Furthermore, a prototype application will be developed using Truffle and Ganache to simulate a blockchain environment and demonstrate the feasibility of the proposed system. The system

will also incorporate Aadhaar Offline Verification, which ensures that voters use an official, digitally signed Aadhaar XML file for authentication, reducing the risk of fake or altered documents. The verification mechanism will confirm authenticity using UIDAI’s PKI-based signature, preventing tampering while maintaining privacy. Additionally, it checks if the voter meets the age requirement without revealing their exact date of birth. This prototype will undergo preliminary testing and evaluation to ensure it meets fundamental security and performance requirements. Finally, the project will conclude with an analysis of the results, identifying potential areas for future improvements.

###### Applicability

The applicability of this project extends to a wide range of scenarios requiring secure and verifiable voting. This includes government elections where transparency and security are of paramount importance, as well as organizational elections within corporations or universities. Furthermore, the system could be applied to any situation demanding secure online polls or community decision- making processes.

###### Achievements

Through this Blockchain-Based Voting System project, we anticipate gaining a comprehensive understanding of blockchain technology and its practical applications in securing electoral processes. We expect to develop proficiency in utilizing development tools such as Truffle and Ganache for building and testing blockchain applications. Furthermore, this project will provide valuable insights into the design and implementation of smart contracts for secure vote recording and counting.

##### ORGANISATION OF REPORT

* **Chapter 1 - Introduction**, this project outlines its goal of developing a secure and transparent voting system using blockchain technology. It highlights blockchain's strengths in addressing vulnerabilities of traditional voting, leading to the development of a prototype for secure vote recording

and counting. The chapter explores blockchain's potential in various voting scenarios, emphasizing the project's aim for a comprehensive understanding of the technology and its application in securing elections.

* **Chapter 2** – **Survey of Technologies**, the details of the technologies necessary to complete the project will be summarized. This includes a comparative study of available technologies and an explanation of the chosen technologies.
* **Chapter 3 – Requirement Analysis,** the problem statement will be defined and divided into sub-problems. Requirement specifications will describe the system components and their functionalities. Planning and scheduling will be illustrated using Gantt charts and PERT, along with hardware and software specifications. Conceptual models, such as Data Flow Diagrams, ER diagrams, and System Flowcharts, will be included.
* **Chapter 4 – System Design,** this chapter describes the desired features and operations in detail, including screen layouts, business rules, process diagrams, basic modules, data design, procedural design with logic diagrams and algorithms, user interface design, security considerations, and test case design.
* **Chapter 5 – Implementation and Testing,** The implementation plan and key coding sections will be outlined with explanations and output screenshots. Code efficiency and optimization techniques will be discussed. Testing will include Unit Testing for individual modules and Integrated Testing for the entire system. Modifications made during testing to resolve issues and improve functionality will also be documented.
* **Chapter 6: Results and Discussion ,**Test reports will summarize the outcomes of various test cases, showing the project's ability to handle different scenarios. Sample inputs and outputs will be provided. The User Documentation will explain the software's functions and components, with screenshots, ensuring users can easily understand and operate the system.
* **Chapter 7: Conclusion**, The overall achievements and outcomes of the project will be summarized. It will reflect on how the project successfully

met its objectives, such as developing an efficient text summarization, paraphrasing, and grammar checking tool. The chapter will also discuss the potential for future improvements and enhancements, emphasizing the project's impact and utility in real-world scenarios. Ultimately, the conclusion will reinforce the significance of the project and its contributions

* **Chapter 8: Cost and Benefit Analysis**, The costs associated with the development and deployment of the project will be outlined, including resources such as software, hardware, and time. This will be followed by an analysis of the benefits, such as improved functionality, increased efficiency, and potential future scalability.

# CHAPTER 2

## SURVEY OF TECHNOLOGY

###### Front-End Features

In developing the front-end of a blockchain-based voting system, several technologies can be considered. **Angular**, **Vue.js**, and **React.js** are popular frameworks for building user interfaces. **Angular** is known for its strong community support and powerful features like two-way data binding and dependency injection, making it a solid choice for complex applications. **Vue.js** offers simplicity and flexibility, with a gentle learning curve, which could be beneficial for developers new to front-end frameworks. **React.js**, on the other hand, is widely adopted due to its component-based architecture and efficiency in rendering user interfaces.

For this project, **Web3** has been chosen as the primary technology for building the user interface. **Web3** integrates seamlessly with **MetaMask,** enabling easy interaction with the Ethereum blockchain. The front-end will facilitate user interactions, allowing them to connect their MetaMask wallets, cast votes, and confirm transactions on the Ethereum blockchain.

###### Back-End Features

When it comes to the back-end, there are multiple blockchain platforms and development tools to consider. **Ethereum, Hyperledge**r Fabric , and Corda are among the top platforms for developing decentralized applications (dApps). **Ethereum** is the most established, with a large developer community and extensive tooling support, making it ideal for deploying smart contracts. **Hyperledger Fabric** offers a permissioned blockchain suitable for enterprise use cases requiring high levels of privacy and control. **Corda** is another enterprise-focused platform, known for its ability to manage complex workflows and transactions.

For this project, **Ethereum** has been selected as the blockchain platform due to its robustness and widespread support for dApp development. The decentralized nature of Ethereum ensures secure and immutable processing of votes. The smart contracts, written in **Solidity**, will manage the voting logic. **Truffle**, a comprehensive development framework, will be employed for writing, testing, and deploying these smart contracts, streamlining the entire development process.

For testing purposes, tools like **Ganache**, **Remix,** and **Hardhat** are commonly used in the Ethereum ecosystem. **Ganache** provides a local blockchain simulation, which is ideal for rapid testing and debugging of smart contracts. **Remix** is a web-based IDE that allows developers to write, compile, and deploy smart contracts directly from the browser.

**Hardhat** is a versatile development environment that offers advanced features like task automation and script running.

In this project**, Ganache** will be utilized as the primary testing environment. It allows for the simulation of the Ethereum blockchain on a local machine, enabling thorough testing of the smart contracts before they are deployed on the main network.

###### Justification - Selected Technologies

The technologies chosen for this blockchain-based voting system— **React.js**, **Ethereum, Solidity, Truffle** and **Ganache**—offer a balanced combination of transparency, immutability, and user accessibility. The Ethereum blockchain ensures that all voting records are secure and cannot be altered post-submission. MetaMask provides an intuitive interface for users to engage with the blockchain, making the voting process accessible to a broader audience. The development and testing tools, Truffle and Ganache, simplify the smart contract lifecycle, ensuring that the project is not only easy to develop and test but also scalable for future enhancements.

# CHAPTER 3

## REQUIREMENTS AND ANALYSIS

##### PROBLEM DEFINITION

The current landscape of voting systems, encompassing both traditional paper-based and electronic methods, is fraught with vulnerabilities that undermine the integrity and trustworthiness of democratic elections. These vulnerabilities manifest in various forms, including security breaches in electronic systems, fraudulent activities such as vote tampering and impersonation, a lack of transparency in vote counting processes, and concerns about centralized control over voting infrastructure. Such shortcomings erode public trust and necessitate the exploration of innovative solutions.

This project aims to leverage the transformative potential of blockchain technology to address these critical challenges by creating a secure, transparent, and tamper-proof voting system. By utilizing a decentralized and immutable ledger, the system will mitigate the risks of manipulation and fraud, ensuring the integrity of the electoral process and fostering greater trust among voters and stakeholders.

##### REQUIREMENT SPECIFICATION

The proposed blockchain-based voting system must meet several crucial requirements to ensure its effectiveness and trustworthiness. These include robust security measures to prevent manipulation and unauthorized access, transparency in vote recording and counting processes, immutability of recorded votes, anonymity to protect voter identities, and accessibility for all eligible voters regardless of technical expertise. Meeting these requirements will establish a foundation for a secure, transparent, and inclusive voting system.

##### HARDWARE AND SOFTWARE REQUIREMENTS

Hardware:

* + - **Processor:** Intel i5 or equivalent
    - **RAM:** 8 GB or higher
    - **Storage:** 100 GB or more
    - **Network:** Stable internet connection

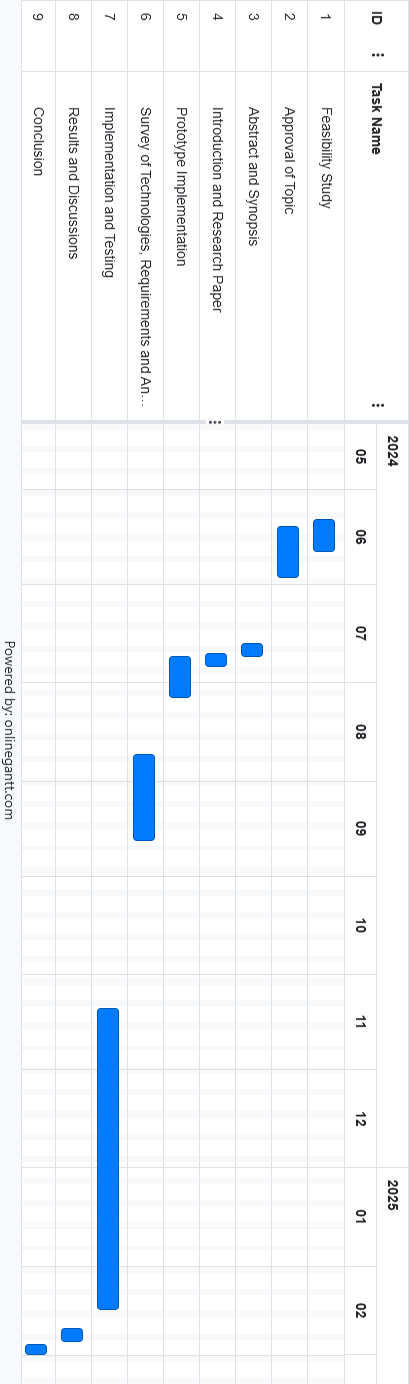
Software:

* **Operating System:** Windows, macOS, or Linux
* **Blockchain Platform:** Ethereum
* **Development Tools:** Truffle Suite, Ganache, MetaMask
* **Programming Languages:** Solidity (for Smart Contracts), JavaScript (for front-end development)
* **Front-End Framework:** React.js (or similar)

##### PLANNING AND SCHEDULING

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Start Date** | **End Date** | **Duration** |
| Feasibility Study | 09-06-2024 | 19-06-2024 | 10 |
| Approval of Topic | 12-06-2024 | 28-06-2024 | 17 |
| Abstract and Synopsis | 19-07-2024 | 19-07-2024 | 1 |
| Introduction and Research Paper | 22-07-2024 | 22-07-2024 | 1 |
| Prototype Implementation | 23-07-2024 | 26-07-2024 | 4 |
| Survey of Technologies, Requirements and Analysis and System Design | 23-08-2024 | 23-08-2024 | 1 |
| Implementation and Testing | 10-11-2024 | 15-02-2025 | 97 |
| Results and Discussions | 20-02-2025 | 23-02-2025 | 3 |
| Conclusion | 25-02-2025 | 28-02-2025 | 4 |
| Documentation | 12-01-2025 | 7-03-2025 | 55 |

* 1. **GANTT CHART**



**Fig 3.1 Gantt Chart**

# CHAPTER 4

## SYSTEM DESIGN

### DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a visual representation of how data moves through a system. It shows how information is input, processed, stored, and output, highlighting the flow and transformation of data within a process.

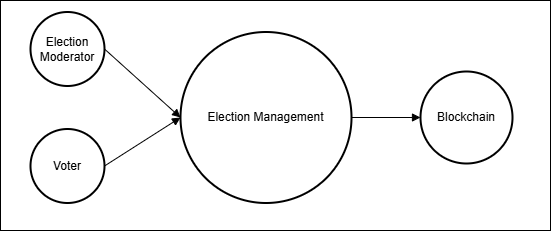
|  |  |  |
| --- | --- | --- |
| Symbol | Name | Description |
|  | **External entity** | External entities are objects outside the system with which system communicates. These are sources and destinations of the system inputs and outputs. |
|  | **Process** | Processes within the system transform data, and data stores hold information for later use. The diagram uses arrows to represent the flow of data between these components, providing a visual representation of the system's operation. |
|  | **Data flow** | Data flow is the path for data to move from one part  of the system to another. It may be a single data element or set of data element. The symbol of data flow is the arrow. The arrow shows the flow  direction. |

### DFD Level 0

**4.**

**1.2 DFD Level 1**

***Table 4.1***

**Figure 4.1 DFD Level 0**

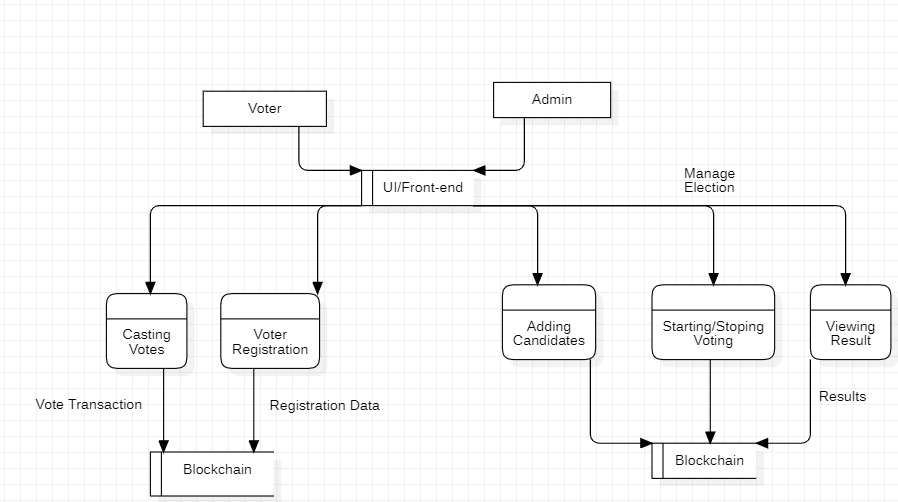


Figure 4.2 DFD Level 1

The voting process begins with the User casting a vote through the Voting Interface. This vote, along with its digital signature, is then passed to the Digital Signature component for verification. Upon successful verification, the Validation process confirms the vote's validity and adds it to the Blockchain for secure and permanent record. The system then processes the votes, making the Results available to both the Public Ledger for transparency and the Query System for data retrieval. An Auditing process ensures the accuracy and integrity of the results, with an Auditor reviewing the audit findings.

### CLASS DIAGRAM

Class diagrams are used to describe the structure of the system. Classes are abstractions that specify the common structure and behaviour of a set of objects. Class diagrams describe the system in terms of objects, classes, attributes, operations and their associations.

|  |  |  |
| --- | --- | --- |
| Symbol | Name | Description |
|  | Class | A blueprint or template that defines the structure and behavior of objects. It encapsulates attributes and operations that  objects of the class can perform. |
|  | Attribute | A characteristic or property of an object. It represents the data held by an object and can have specific  data types. |
| UML Class Diagram Notation - Operation | Operation | An action or behaviour that an object can perform. It defines the functionality associated with a  class. |
| UML Class Diagram Notation - Responsibility | Responsibility | A high-level obligation or contract that a class fulfills. It defines the overall purpose and role of the  class. |
|  | Association | A link or connection between two classes, indicating a relationship between them. |

***Table 4.2***

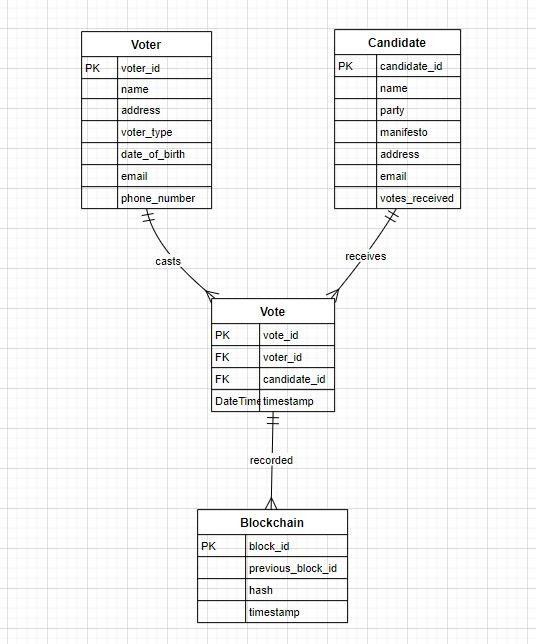


Figure 4.3 CLASS DIAGRAM

In the voting system, each Voter casts a single Vote linked to a specific Candidate in an Election. All Vote transactions are recorded on the Blockchain for security and verifiability. This reflects the 1-to-Many relationships between Voter and Vote, Candidate and Vote, and Blockchain and Vote.

###### Use Case Diagram

A use case diagram is a visual representation of the interactions between users (actors) and a system. It shows the different functionalities offered by the system and how users can interact

|  |  |  |
| --- | --- | --- |
| Symbol | Name | Description |
|  | Use Case | Represents a specific action or goal that a user can achieve by interacting with the system. It defines a complete unit of functionality. |
|  | Actor | Represents a user or external entity that interacts with the system. It can be a person, another system, or a device. |

with them to achieve their goals.

***Table 4.3***

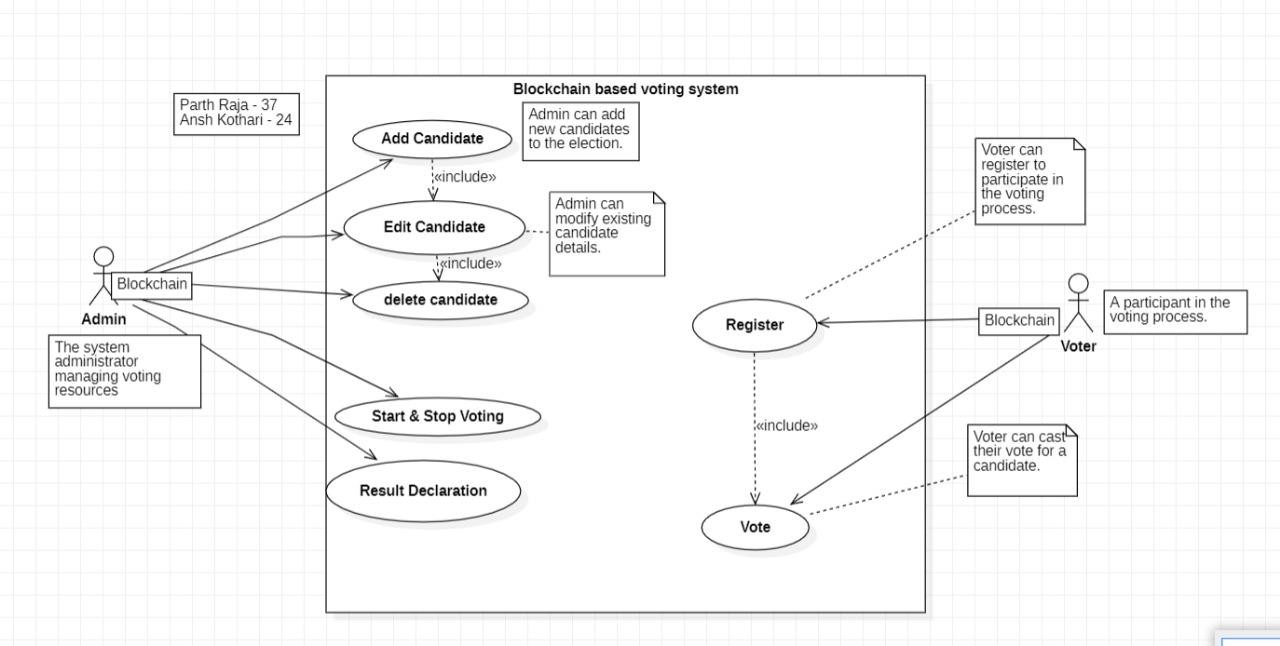


Figure 4.4 Use Case Diagram

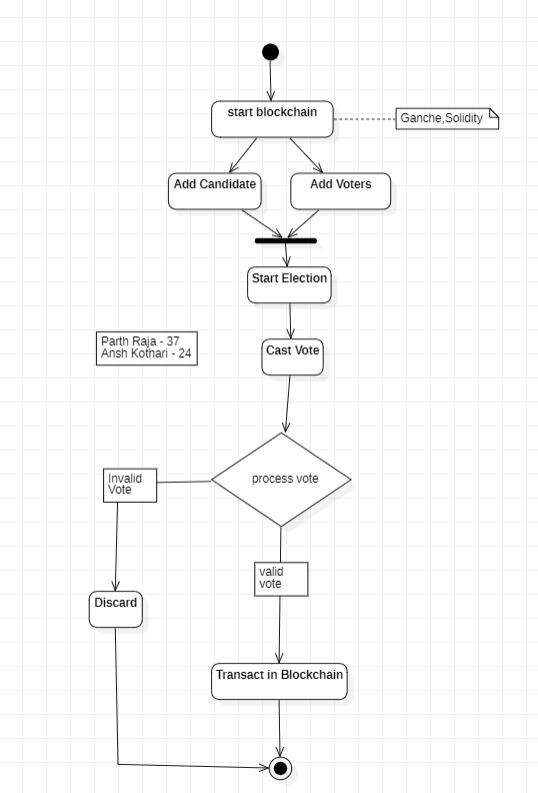
The use case diagram depicts the roles and actions of key participants in a blockchain voting system. Voters cast votes, verify them, register, authenticate, and edit profiles. Election commissioners monitor ballots and check tallies. System admins handle election setup and voter registration. This diagram visualizes the system's functionality and how different actors interact to ensure secure and transparent elections.

##### ACTIVITY DIAGRAM

An activity diagram visually represents the flow of activities and actions within a process or system. It uses symbols to depict different types of activities, decision points, and the sequence of execution, providing a clear overview of the workflow.

|  |  |  |
| --- | --- | --- |
| Symbol | Name | Description |
|  | Start symbol | Initiates the flow of activities in an activity diagram. |
|  | Activity Symbol | Represents a specific task or action performed within the workflow. |
|  | Control Flow | It shows the directional flow, or control flow, of the activity. An arrow indicating the order in which activities are executed. |
|  | Joint symbol | Combines separate paths of execution back into a single flow.Represented with a thick vertical or horizontal line. |
|  | Fork Symbol | Divides a single flow into parallel paths that can execute simultaneously. Symbolized with multiple arrowed lines from a join. |
|  | Decision Symbol | Indicates a point where the flow branches based on a condition or decision. It represents the branching or merging of various flows  with the symbol acting as a frame or container. |

***Table 4.4***



***Figure 4.5 ACTIVITY DIAGRAM***

# CHAPTER 5

#### IMPLEMENTATION AND TESTING

###### Implementation Approach

Our approach to implementing the Blockchain Voting System was driven by the need for a system that was not only secure and transparent but also scalable and easy to maintain. We opted for a modular architecture, breaking down the system into key functional units. Solidity and Ethereum were the natural choices for the smart contract development, given Ethereum's maturity and the robust tooling available. Truffle Suite became our central hub for development, testing, and deployment, streamlining the entire blockchain application lifecycle.

The core of our system is structured around these modules:

1. **Voting Smart Contract (`Voting.sol`):** This Solidity contract is the backbone, housing all the core logic and data for the voting process. We carefully designed it to manage candidates, voters, and election administrators. Efficiency was a key concern, so we used `mappings` for quick data lookups and `events` for efficient off-chain tracking of important actions.
2. **Voting Lifecycle Management:** This module, implemented within the smart contract, governs the entire election timeline. It includes functions for administrators to manage candidates (adding, editing, deleting), to control voter registration, to initiate and conclude the voting period, and of course, the crucial function for voters to cast their ballots.
3. **Secure Access Control**: Security is paramount in a voting system. We implemented robust access control using Solidity `modifiers`. Critical administrative functions are strictly protected, ensuring that only authorized administrators can perform actions like setting up the election or viewing results.
   1. Coding Details and Efficiency:

pragma solidity ^0.8.19;

contract Voting { struct Candidate {

uint id; string name;

string politicalParty; string description; string imageURL;

string constituencyPincode; uint voteCount;

bool exists;

}

mapping(uint => Candidate) public candidates; mapping(address => bool) public voters; mapping(address => bool) public registeredVoters; mapping(address => bool) public hasVoted; mapping(address => bool) public admins;

uint public candidatesCount; bool public votingActive; address public initialAdmin;

event CandidateAdded(uint id, string name, string politicalParty, string description, string imageURL, string constituencyPincode);

event CandidateEdited(uint id, string newName, string newPoliticalParty, string newDescription, string newImageURL, string newConstituencyPincode);

event CandidateDeleted(uint id);

event AdminRegistered(address newAdmin); event VoterRegistered(address voter); constructor() {

initialAdmin = msg.sender; admins[initialAdmin] = true; candidatesCount = 0; votingActive = false;

}

modifier onlyAdmin() {

require(admins[msg.sender], "Only admin can perform this action");

\_;

}

function registerAdmin(address \_newAdmin) public onlyAdmin { require(\_newAdmin != address(0), "Invalid address");

require(!admins[\_newAdmin], "Admin already registered"); admins[\_newAdmin] = true;

emit AdminRegistered(\_newAdmin);

}

function addCandidate( string memory \_name,

string memory \_politicalParty, string memory \_description, string memory \_imageURL,

string memory \_constituencyPincode

) public onlyAdmin {

require(bytes(\_name).length > 0, "Candidate name cannot be empty"); candidatesCount++;

candidates[candidatesCount] = Candidate( candidatesCount,

\_name,

\_politicalParty,

\_description,

\_imageURL,

\_constituencyPincode,

0,

true

);

emit CandidateAdded(candidatesCount, \_name, \_politicalParty, \_description,

\_imageURL, \_constituencyPincode);

}

function editCandidate( uint \_candidateId,

string memory \_newName,

string memory \_newPoliticalParty, string memory \_newDescription, string memory \_newImageURL,

string memory \_newConstituencyPincode

) public onlyAdmin {

require(\_candidateId > 0 && \_candidateId <= candidatesCount, "Invalid candidate ID");

require(candidates[\_candidateId].exists, "Candidate does not exist"); require(bytes(\_newName).length > 0, "New name cannot be empty"); candidates[\_candidateId].name = \_newName; candidates[\_candidateId].politicalParty = \_newPoliticalParty; candidates[\_candidateId].description = \_newDescription;

candidates[\_candidateId].imageURL = \_newImageURL;

candidates[\_candidateId].constituencyPincode = \_newConstituencyPincode; emit CandidateEdited(\_candidateId, \_newName, \_newPoliticalParty,

\_newDescription, \_newImageURL, \_newConstituencyPincode);

}

function deleteCandidate(uint \_candidateId) public onlyAdmin {

require(\_candidateId > 0 && \_candidateId <= candidatesCount, "Invalid candidate ID");

require(candidates[\_candidateId].exists, "Candidate already deleted"); candidates[\_candidateId].exists = false;

emit CandidateDeleted(\_candidateId);

}

function registerVoter() public { require(!voters[msg.sender], "Voter already registered"); voters[msg.sender] = true;

registeredVoters[msg.sender] = true; emit VoterRegistered(msg.sender);

}

function startVoting() public onlyAdmin { votingActive = true;

}

function endVoting() public onlyAdmin { votingActive = false;

}

function vote(uint \_candidateId) public { require(votingActive, "Voting is not active");

require(registeredVoters[msg.sender], "You must be registered to vote"); require(!hasVoted[msg.sender], "You have already voted");

require(\_candidateId > 0 && \_candidateId <= candidatesCount, "Invalid candidate");

require(candidates[\_candidateId].exists, "Candidate does not exist");

candidates[\_candidateId].voteCount++; hasVoted[msg.sender] = true;

}

function getTotalVotes(uint \_candidateId) public view returns (uint) {

require(\_candidateId > 0 && \_candidateId <= candidatesCount, "Invalid candidate");

require(candidates[\_candidateId].exists, "Candidate does not exist"); return candidates[\_candidateId].voteCount;

}

function getAllCandidates() public view returns (string[] memory, uint256[] memory) {

string[] memory names = new string[](candidatesCount); uint256[] memory votes = new uint256[](candidatesCount); for (uint i = 1; i <= candidatesCount; i++) {

if (candidates[i].exists) {

names[i - 1] = candidates[i].name; votes[i - 1] = candidates[i].voteCount;

}

}

return (names, votes);

}

}

###### Code Efficiency

Writing efficient Solidity code was a priority, especially considering Ethereum's gas costs. We employed several strategies to optimize `Voting.sol` for both gas efficiency and code readability:

1. **Efficient Data Structures with Mappings:** We strategically used

`mappings` for storing and retrieving data like candidates and voters. Mappings offer constant-time lookups, significantly outperforming arrays in terms of gas consumption, especially as the dataset grows.

1. **Event Logging for Off-Chain Monitoring:** We made extensive use of

`events` to log key actions happening within the contract, such as voter registration, vote casting, and administrative actions. Events are a gas-efficient way to make important information available off-chain for our front-end and auditing purposes, without incurring the higher costs of storing everything directly on the blockchain's state.

1. **`require` Statements for Early Error Handling:** To prevent wasted gas on invalid operations, we implemented `require` statements at the beginning of functions to check preconditions. This ensures that if a condition is not met (e.g., voting not active, voter already voted), the function execution halts immediately, saving gas and providing informative error messages.
2. **Modifiers for Reusable Access Control:** We encapsulated access control logic within Solidity `modifiers` like `onlyAdmin`. This promotes code reusability, improves readability, and reduces the chance of errors by centralizing access control checks, rather than repeating the same logic in multiple functions.

###### Testing Approach:

Given the critical nature of a voting system, we adopted a comprehensive, multi- stage testing approach. This included unit testing, integrated testing, and beta testing, each designed to validate different aspects of the system.

* + 1. Unit Testing

Our unit testing phase concentrated on verifying individual functions within the

`Voting.sol` smart contract in isolation. Using Truffle and Ganache, we created a controlled local blockchain environment to test each function thoroughly. Key areas of focus included:

* + - * **Aadhaar XML Verification Logic (Simulated):** While we couldn't directly test Aadhaar XML verification on-chain in the unit tests, we simulated the verification process. We created test scenarios to ensure that our smart contract logic correctly handled the outcome of **Aadhaar XML verification**. For example, we tested cases where the simulated Aadhaar verification was successful (voter registered), and cases where it failed (registration rejected), ensuring the contract behaved as expected in both scenarios based on the verification result. We also tested the **pincode extraction** logic in isolation to confirm it correctly retrieved the pincode from the simulated Aadhaar data.
      * **Voter Registration Flow:** We tested the complete voter registration function, ensuring it correctly registered valid voters. Crucially, we specifically tested the duplicate registration prevention – verifying that attempts to register the same voter multiple times were correctly rejected by the contract.
      * **Candidate Management Functions**: We rigorously tested the admin functions for adding, editing, and deleting candidates. This included testing adding valid candidate data, attempting to add invalid data, editing candidate information, and confirming that deleting a candidate effectively removed them from the voting pool.
      * **Voting Lifecycle Management:** We tested the **`startVoting`** and

**`endVoting`** functions, verifying that they correctly transitioned the voting state and enforced the voting period. We ensured that voting was only possible when the voting state was active and disabled otherwise.

* + - * **Vote Casting and Pincode-Based Candidate Filtering:** We tested the core `castVote` function. We verified that registered voters could successfully cast votes, that voters could only vote once, and that votes were accurately recorded against the selected candidate. Importantly, we tested the pincode-based candidate filtering logic. In our unit tests, we simulated different voter pincodes and verified that when a voter cast a vote, only candidates associated with that pincode (or "all constituencies" candidates) were available for selection.

###### Integrated Testing:

Beyond individual functions, we conducted integrated testing to ensure all modules of the system worked seamlessly together. These tests focused on end-to-end flows, simulating complete user journeys:

1. **Full Voter Registration and Voting Flow (including Aadhaar Simulation):** We tested the entire registration and voting process, simulating a voter starting with registration (including the simulated Aadhaar XML verification step), then logging in, viewing pincode-filtered candidates, casting a vote, and receiving confirmation. This tested the integration between the front-end (simulated in our tests) and the smart contract for the complete voter journey.
2. **Admin Election Setup and Management**: We tested the complete admin workflow, from admin login and candidate addition, to starting the voting process, monitoring progress (simulated), ending the voting, and accessing the final results. This verified the correct functioning of all admin-privileged functions and their impact on the overall system state.
3. **Security and Access Control Enforcement:** We specifically tested the access control mechanisms. We attempted to bypass admin restrictions by trying to call admin-only functions from non-admin accounts, verifying that our

`onlyAdmin` modifier and other access control measures were effectively preventing unauthorized actions.

###### Beta Testing:

To evaluate the system under more realistic conditions, we conducted beta testing in a simulated network environment. This phase aimed to identify any issues that might arise with concurrent users and under stress:

1. **Concurrent Voter Load Simulation:**We simulated multiple voters registering and voting simultaneously to assess the contract's performance under load and identify potential bottlenecks or gas-related issues when many transactions occur concurrently.
2. **Edge Case and Error Handling Stress Testing**: We pushed the system with edge cases and invalid inputs. We simulated scenarios like voters attempting to

vote for non-existent candidates, registering with incomplete or invalid data, or attempting to vote outside the active voting period, to ensure robust error handling and system stability under stress.

1. **End-to-End Election Lifecycle Simulation:** We ran complete simulated elections from start to finish, mimicking a real election scenario as closely as possible. This involved an admin setting up the election, voters registering, voting period starting, voters casting votes, voting period ending, and results being finalized and viewed, providing a holistic test of the entire system lifecycle.

###### Modifications and Improvements Based on Testing

Testing proved invaluable in refining our system. Based on the findings, we made the following key improvements:

1. **Improved Clarity of Error Messages:** We enhanced the error messages in our

`require` statements to be more descriptive and user-friendly. This made debugging easier during development and would provide clearer feedback to users in case of errors in a real-world scenario.

1. **Strengthened Input Validation:** We added more rigorous input validation checks, particularly for voter and candidate data, to prevent malformed or invalid data from entering the system, enhancing overall robustness.

These iterative modifications, directly informed by our testing process, led to a significantly more robust, user-friendly, and efficient Blockchain Voting System.

* 1. **Test Cases**

The following table summarizes key test cases:

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Description** | **Expected Outcome** |
| Add Candidate | Admin adds a valid candidate. | Candidate is successfully added. |
| Edit Candidate | Admin edits an existing candidate’s details. | Candidate details are updated. |

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Description** | **Expected Outcome** |
| Delete Candidate | Admin deletes a candidate. | Candidate is marked as non-existent. |
| Register Voter | A user registers to vote. | Voter registration is successful. |
| Duplicate Voter Registration | A user attempts to register twice. | Error is thrown: “Voter already registered.” |
| Start Voting | Admin starts the voting process. | Voting state is set to active. |
| Cast Vote | A registered voter casts a vote. | Vote is recorded. |
| Double Voting | A voter attempts to vote twice. | Error is thrown: “You have already voted.” |
| End Voting | Admin ends the voting process. | Voting state is set to inactive. |

The above test cases were thoroughly executed, and the results confirmed the robustness of the implementation.

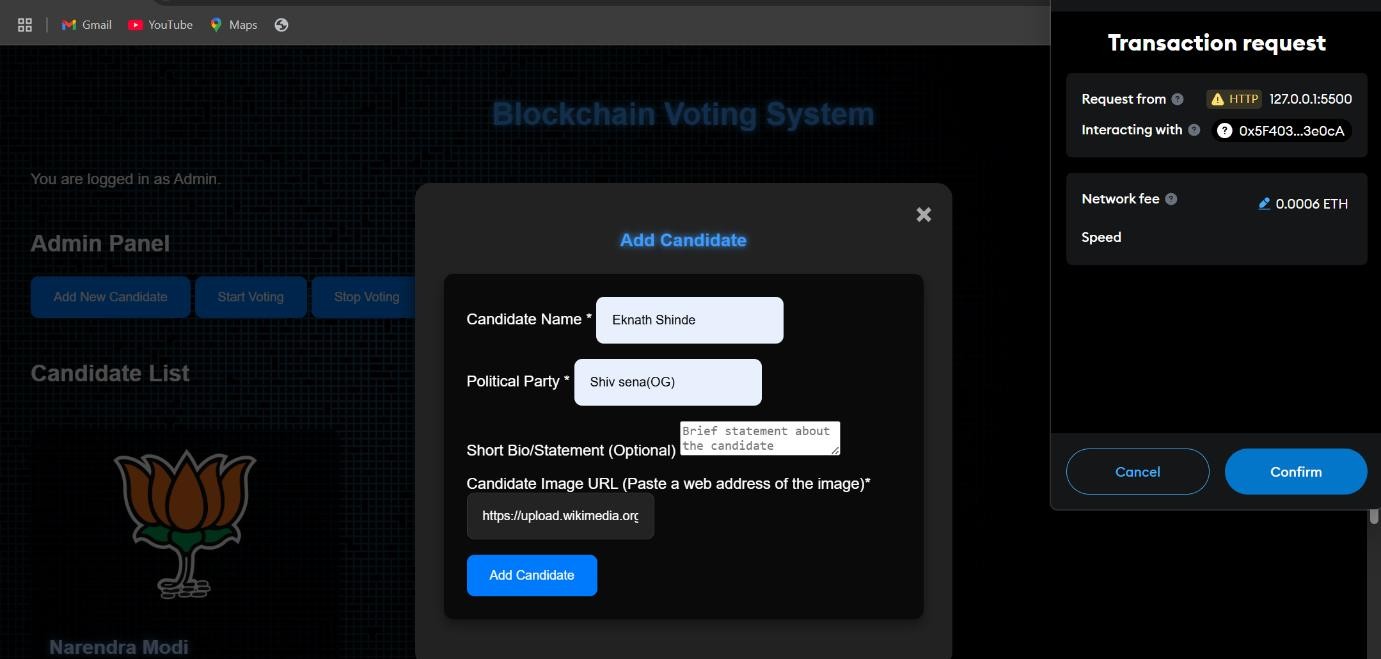
# CHAPTER 6

#### RESULTS AND DISCUSSION

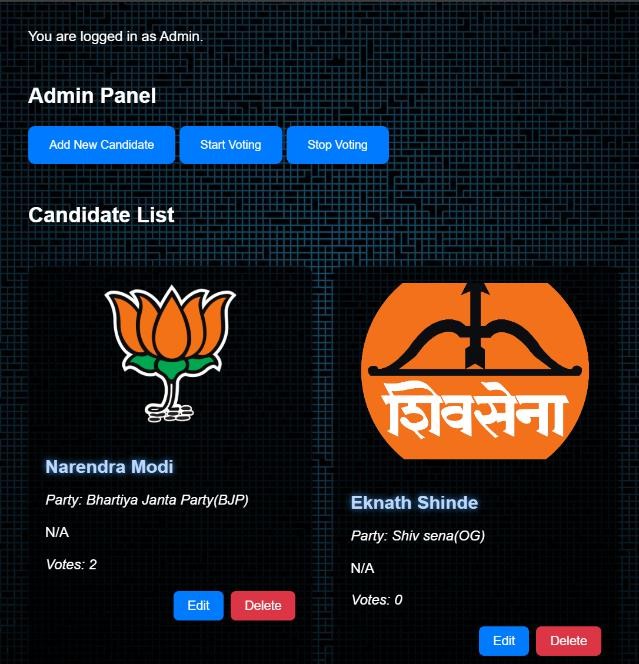
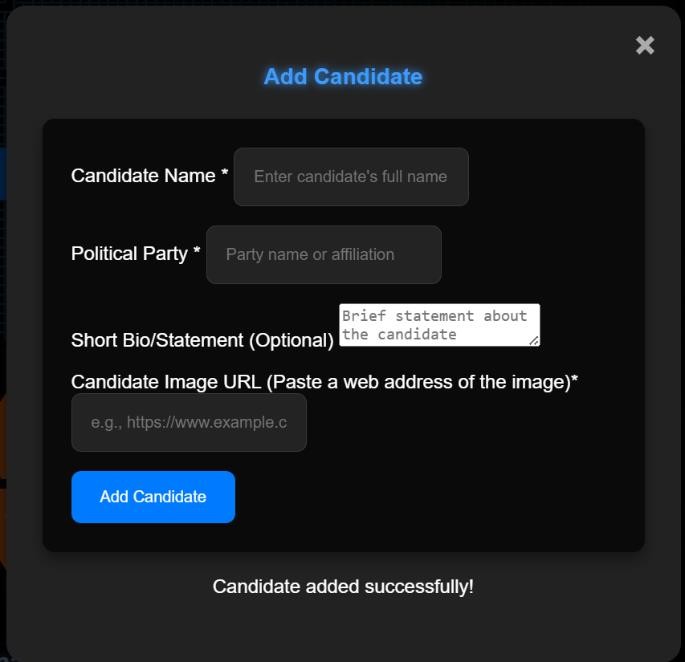
##### TEST REPORT

###### Adding Candidates

Adding candidates can only be performed by Admins. It requires fields like **“Candidate name”, “Political Party”, “Short Description(optional)”, “Political Party image URL” and “Constituency pincode”.** A candidate was successfully added.



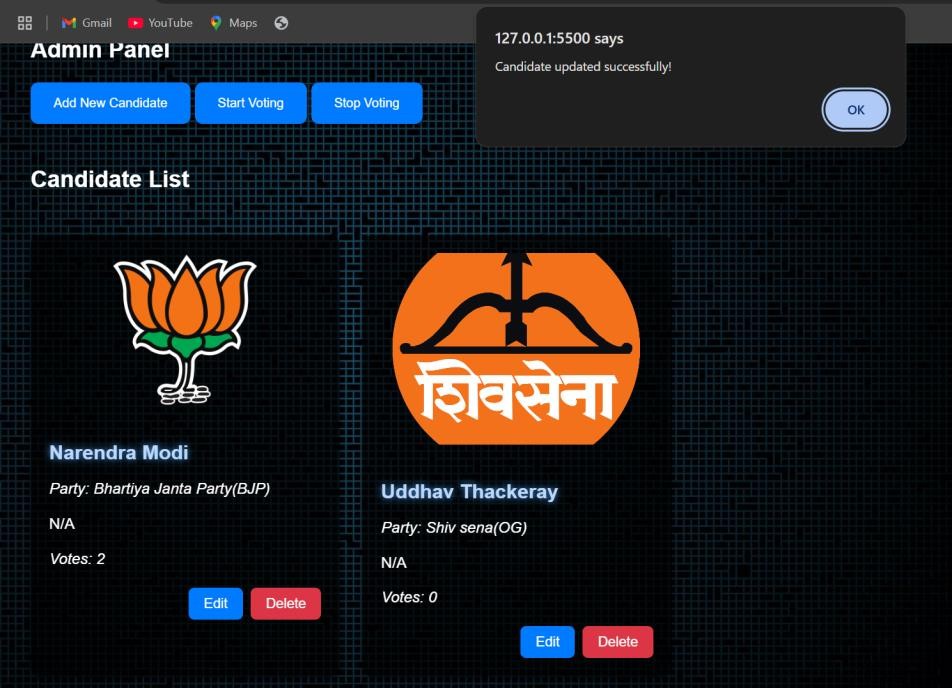
***Figure 6.1***

******

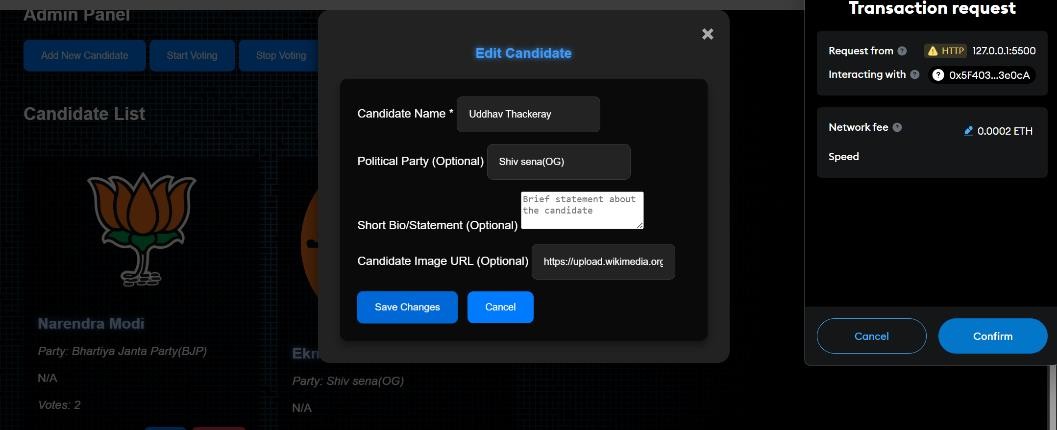
***Figure 6.2 Figure 6.3***

###### Editing Candidates

Editing candidates can only be performed by Admins. It requires fields like **“Candidate name”, “Political Party”, “Short Description(optional)”, “Political Party image URL” and “Constituency pincode”.** A candidate name was successfully edited



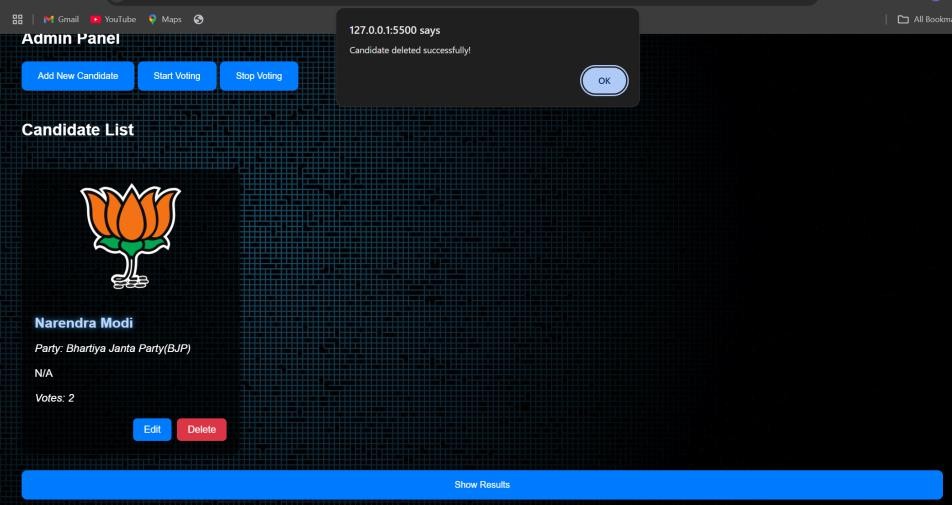
***Figure 6.4***

******

***Figure 6.5***

* + 1. Delete Candidate

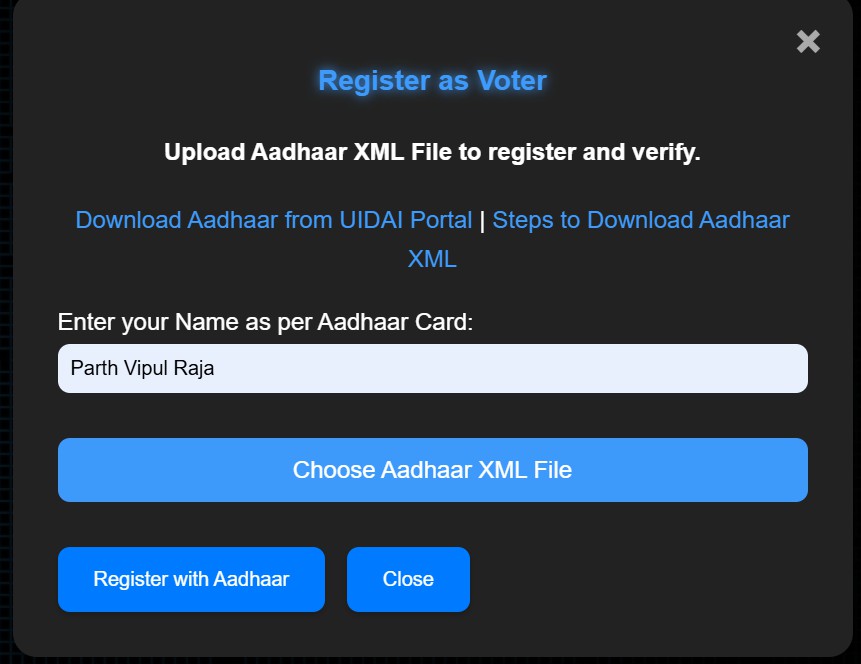
Deleting a candidates can only be performed by Admins. A candidate was deleted successfully.



***Figure 6.6***

###### Register Voter

Voter registration requires **“Aadhar XML file”,** to verify voter’s **age, name and extract pincode**

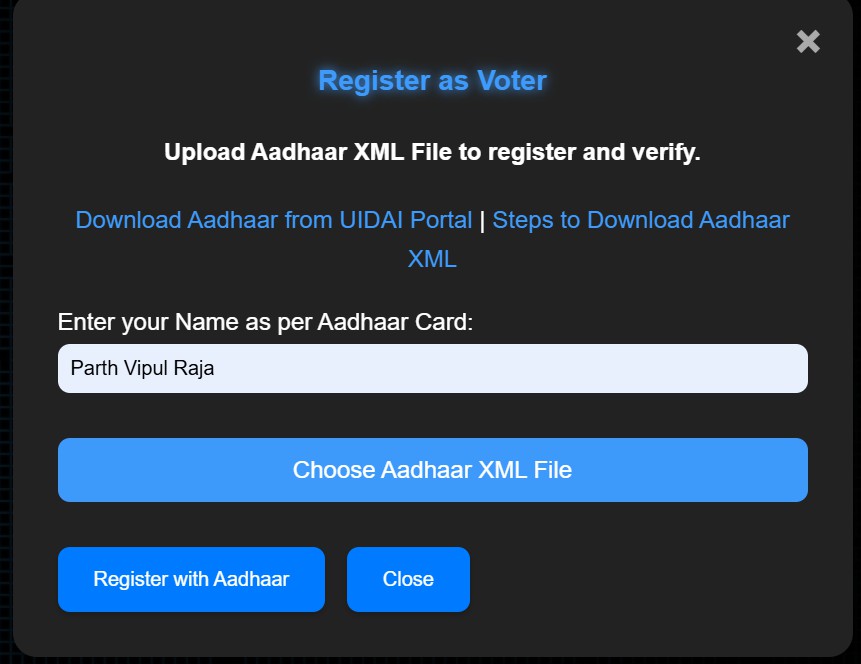
****

***Figure 6.7***

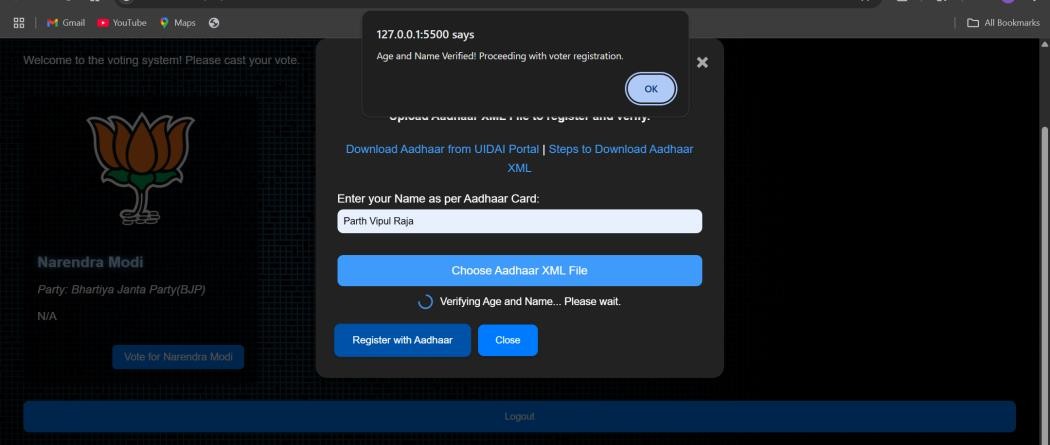
******

***Figure 6.8***

###### Voter’s Aadhar-XML based Age and Name Verification

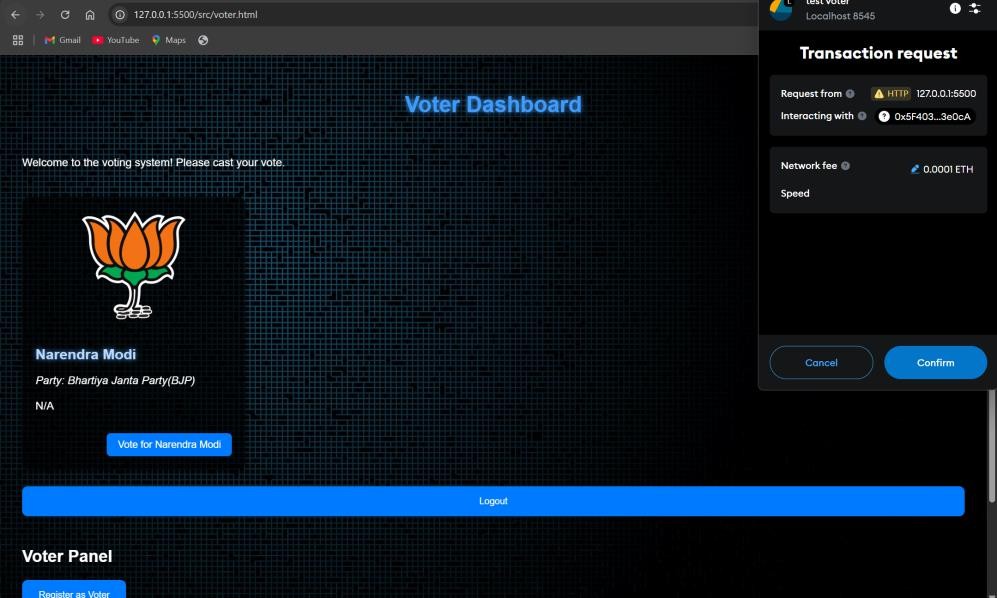
****

***Figure 6.9***

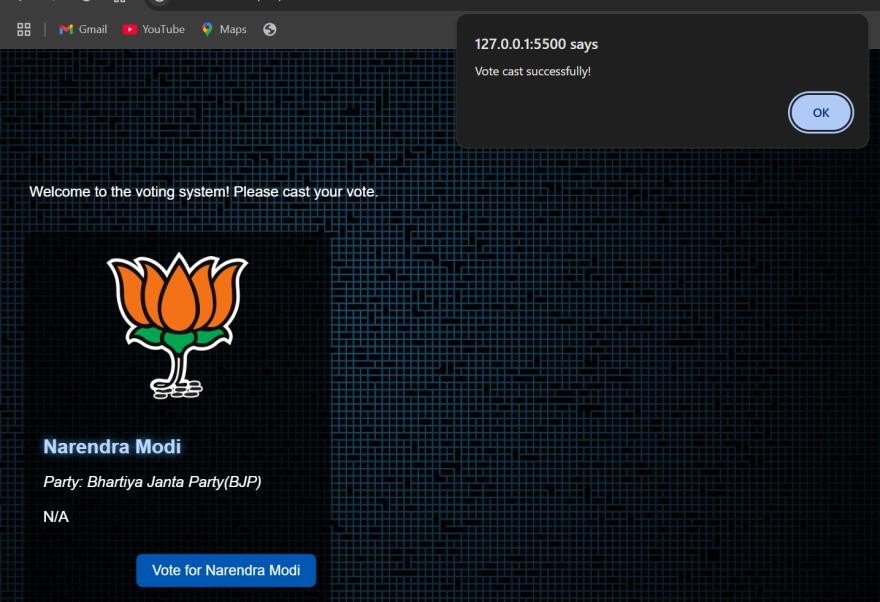
******

***Figure 6.10***

###### Cast Vote

****

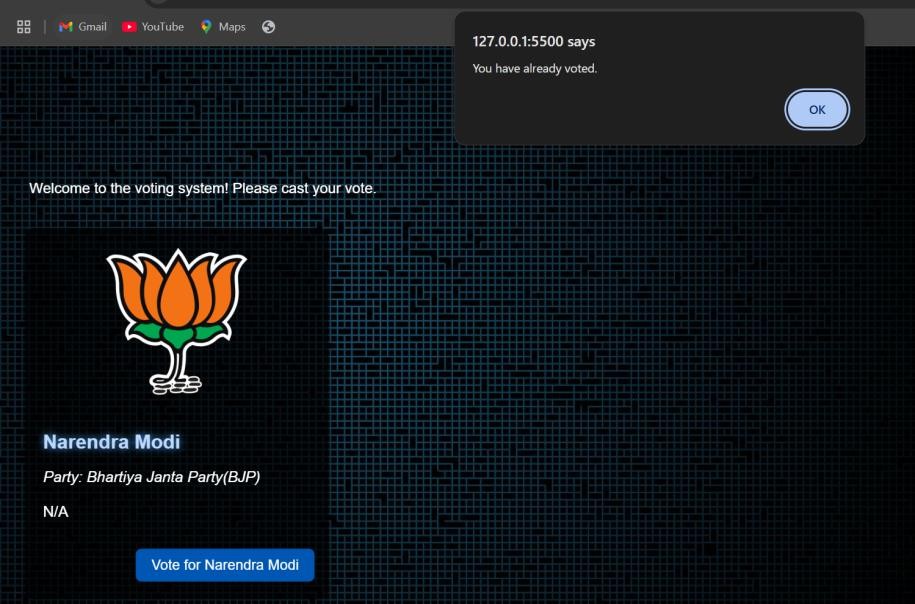
***Figure 6.11***

******

***Figure 6.12***

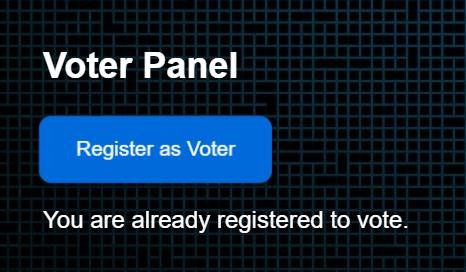
* + 1. Double Voting

A registered voter cannot vote again.



***Figure 6.13***

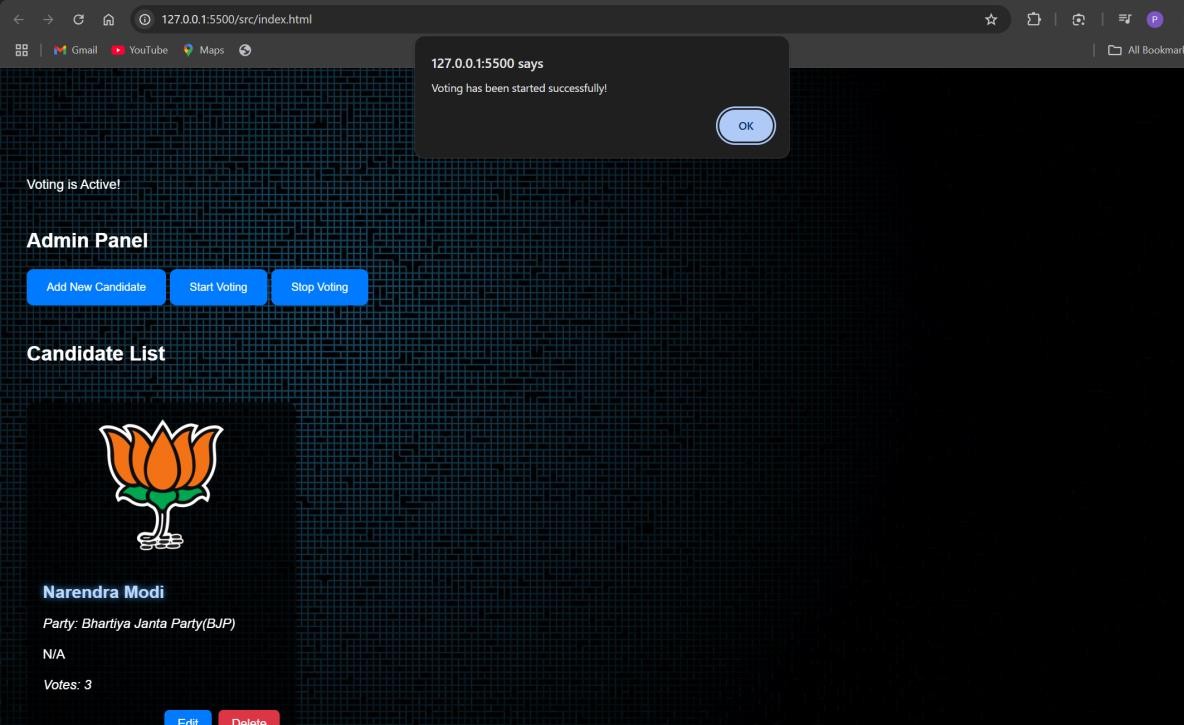
###### Duplicate Voter Registration

****

***Figure 6.14***

* + 1. Start Voting

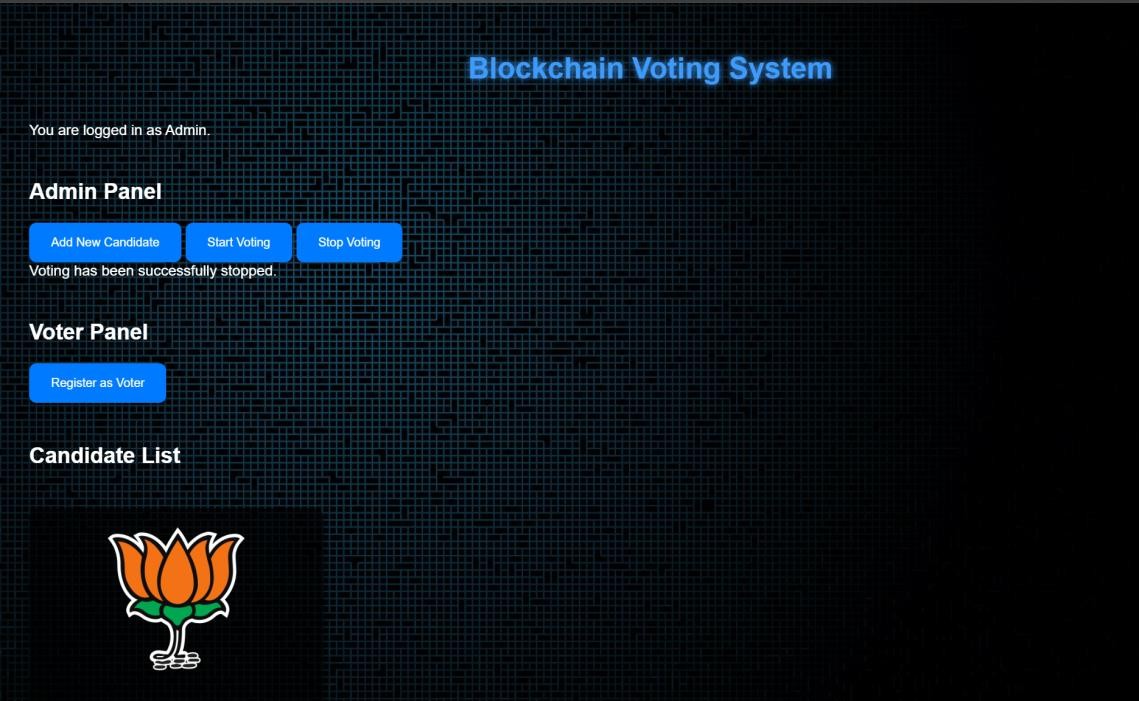
Only admin can start voting. Voter can vote only after voting is started



***Figure 6.15***

* + 1. Stop Voting

**Only admin can stop voting. Voter cannot vote once it is stopped**

****

***Figure 6.16***

###### Results

This user-friendly Blockchain Voting System modernizes elections while prioritizing security and voter relevance. **For voters, functionalities include:** streamlined registration using **Aadhaar Offline XML for name and age verification**, and a simplified ballot showing only candidates in their constituency, **determined by pincode extraction from their Aadhaar XML.** Secure vote casting and transparent results are seamlessly integrated.

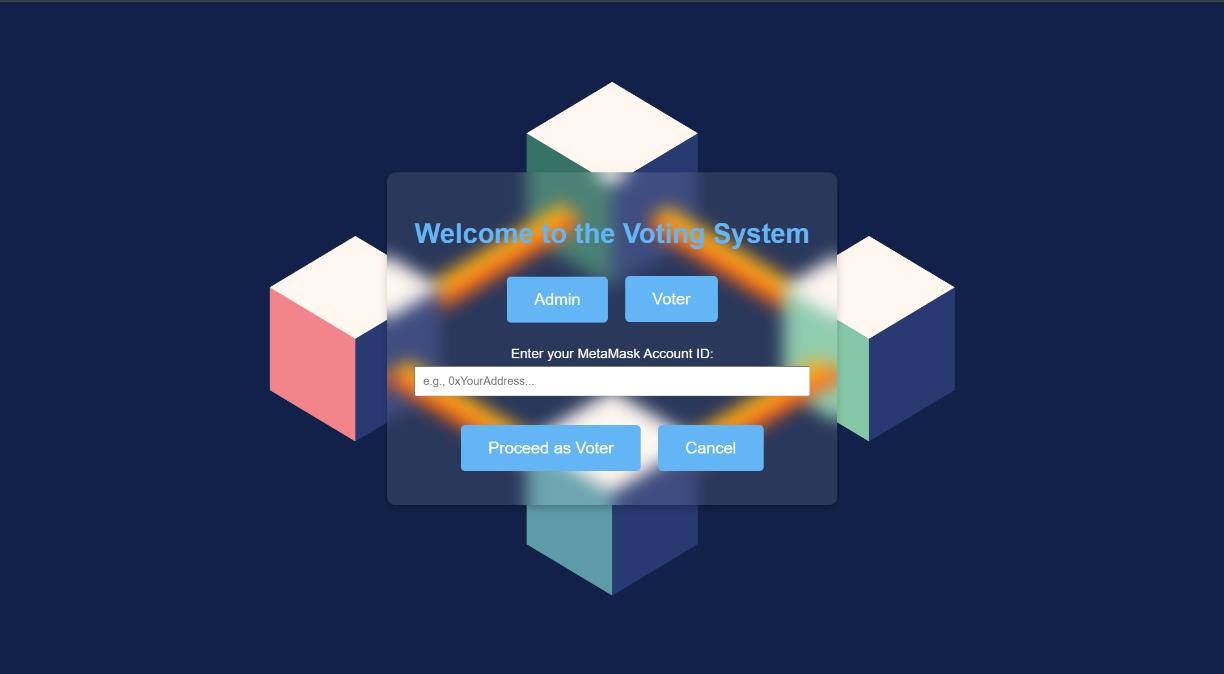
1. Home Page



Asks if user is Admin or Voter

**Figure 6.17**

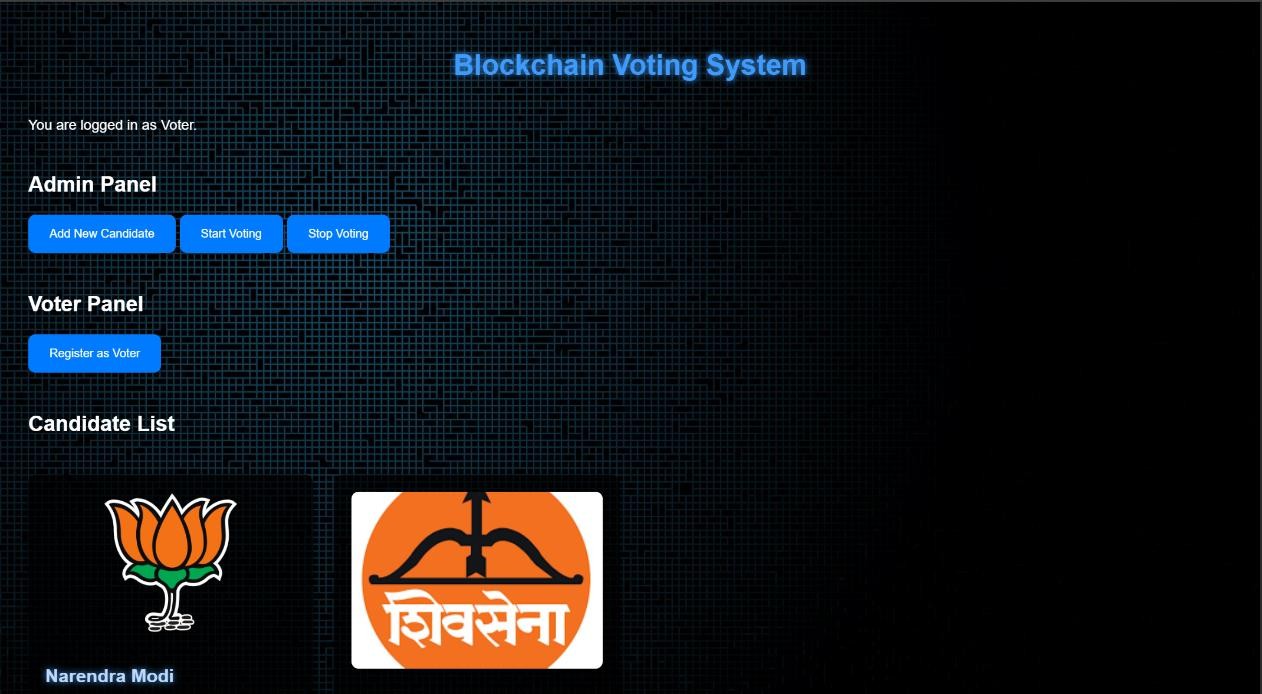
###### Voter Registration(Requires MetaMask Account No.)



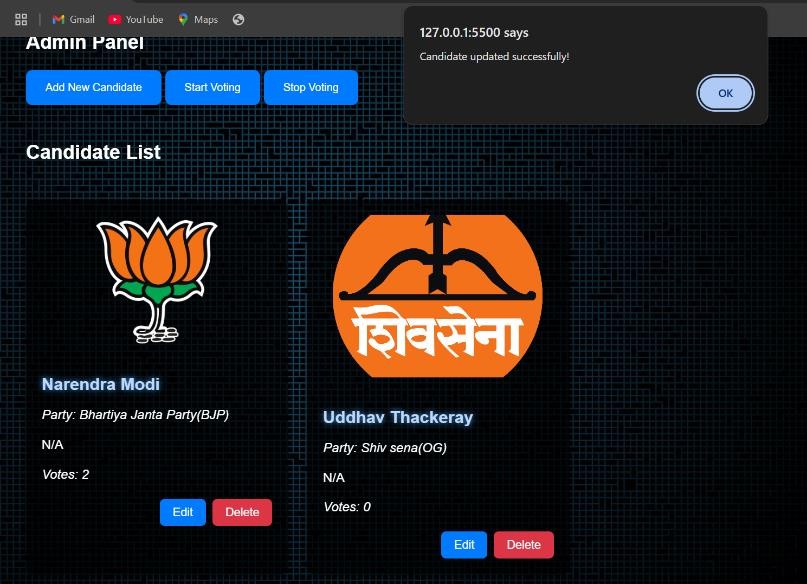
**Figure 6.18**

###### Admin Panel

Allows admin to add, edit and delete candidate. Also allows controls to start and Stop Voting as well as view resuls.



**Figure 6.19**



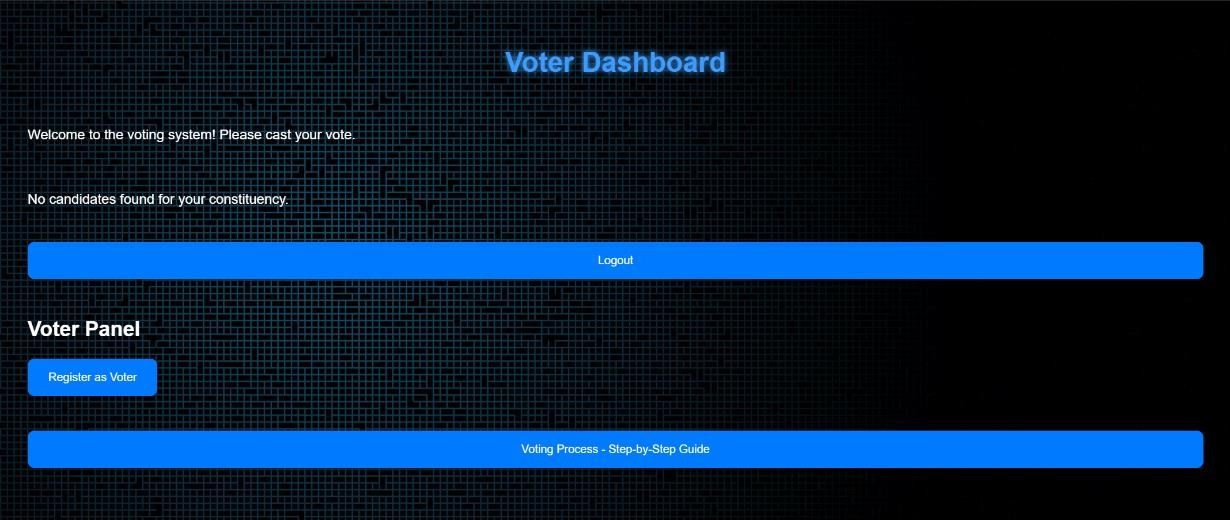
**Figure 6.20**

1. **View Results**

****

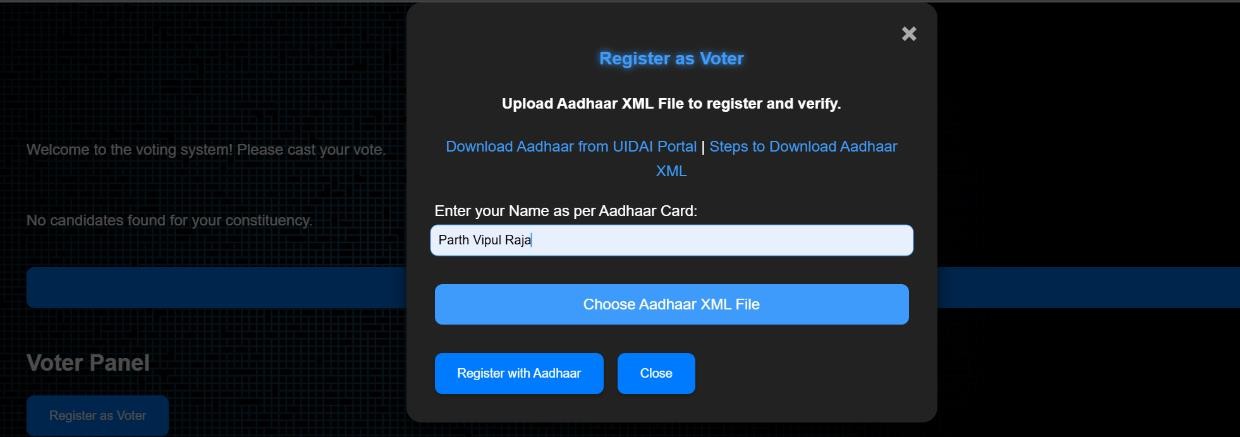
**Figure 6.21**

###### Voter’s Dashboard

****

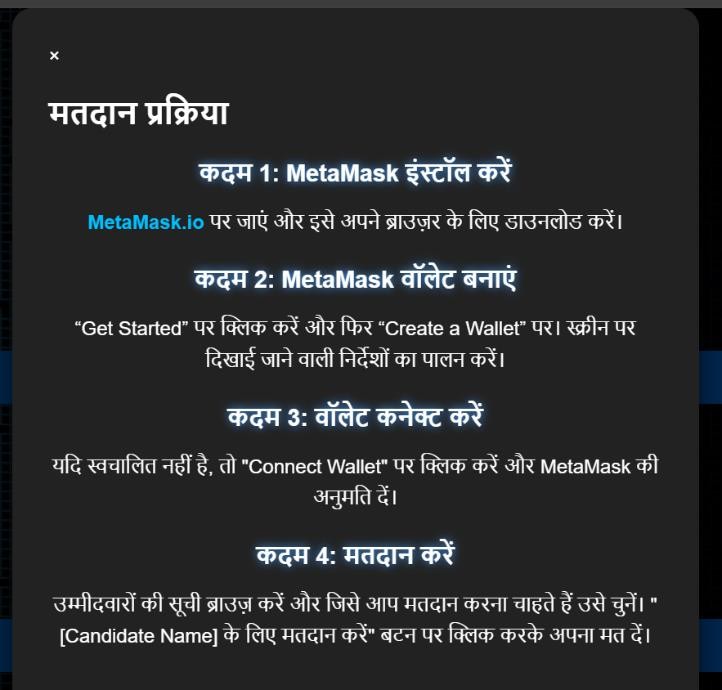
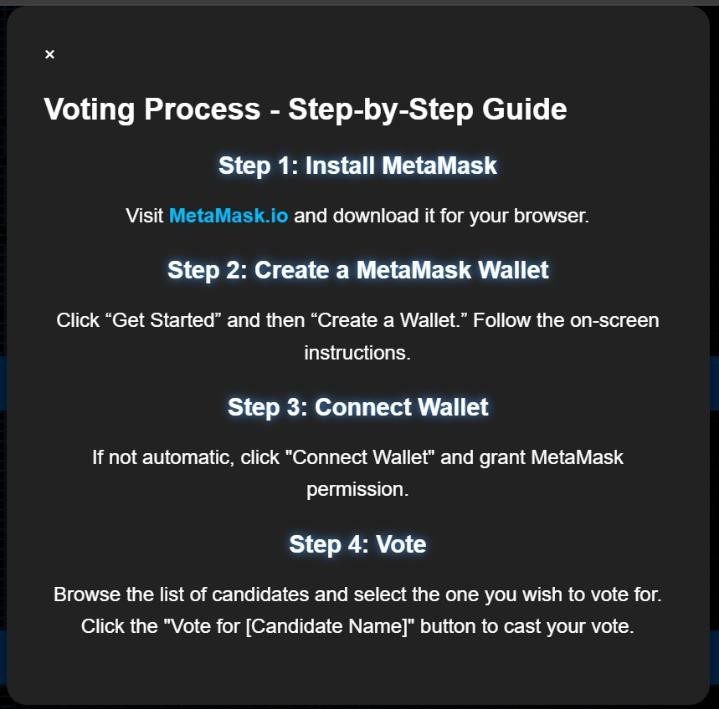
**Figure 6.22**

###### Register voter

****

**Figure 6.23**

1. **Voting Process Step-Step Guide(Hindi/Englsih)**

****

# CHAPTER 7

#### Conclusion

* 1. **Conclusion**

Blockchain-based voting systems have the potential to revolutionize electoral processes by enhancing security, transparency, and trust. By leveraging decentralized ledger technology, cryptographic security, and smart contracts, this system ensures tamper-proof and verifiable voting. The integration of Aadhaar-based voter verification adds an extra layer of authentication, reducing fraudulent activities and improving voter legitimacy.

Through rigorous testing and implementation, this project demonstrates the feasibility of blockchain in elections, providing a modern alternative to traditional voting methods.

While challenges such as scalability and accessibility remain, the system lays a strong foundation for secure digital voting in the future.

###### Future Scope

The future benefits and scope of blockchain-based voting systems are vast, offering numerous opportunities for advancement and widespread adoption:

* + - **Scalability Enhancements:** As blockchain technology evolves, the system can be optimized to handle large-scale elections efficiently. Enhancing transaction speeds and reducing network congestion will be key to ensuring smooth electoral processes.
    - **Improved Voter Accessibility:** Future developments can focus on making the system more user-friendly by introducing mobile-based voting, multilingual support, and a simplified interface to accommodate voters from diverse demographics and technical backgrounds.
    - **Regulatory Compliance:** To facilitate real-world adoption, blockchain voting systems must align with existing electoral laws and regulations. Collaborating with government bodies to establish a legally compliant framework will be essential for integration into official elections.
    - **Enhanced Security Measures:** As cyber threats become more sophisticated, implementing advanced cryptographic techniques, AI-driven fraud detection, and multi-factor authentication will further strengthen voter privacy and election integrity.
    - **Integration with Emerging Technologies:** Incorporating decentralized identity solutions, biometric verification, and zero-knowledge proofs can enhance voter authentication and data security. Additionally, integrating with government databases can streamline the verification process while ensuring privacy.
    - **Global Adoption and Research:** Expanding the blockchain voting model for use in corporate elections, community decision-making, and international governance can broaden its impact. Continuous research and development will drive innovations that make blockchain voting a mainstream reality.

# CHAPTER 8

#### COST AND BENEFIT ANALYSIS

The term **“Cost Benefit Analysis”** refers to how the software will prove to be beneficial than the existing system used in the organization implementing it. Cost Benefit Analysis involves the balance of project expenditure and returns from its implementations. It must be noted that any software implementation involves initial investments and benefits are usually long term. System implementation time may vary for different organizations and it may also take some time for the employees of the different organizations to adjust with the system. A thorough analysis of the system was undertaken by keeping in the cost benefit ratio, details of which are as follows:

**COST EVALUATION**

###### Software Costs:

* + Since Windows, VS Code are freely available, there are no direct software costs associated with acquiring these tools.

###### Analysis and Feasibility Study:

Cost = No. of people involved \* Charges per day \* No. of days

= 2\*200\*10

= Rs. 4,000

###### Development Costs:

* + No major expenses were involved in developing the UI/UX.
  + However, there were personnel costs associated with development, calculated as follows:

No. of people involved in project = 2 Charges per day = 200

No. of days = 114

Developer cost = No. of people involved \* Charges per day \* No. of days

= 2\*200\*114

= Rs. 45,600

###### Domain Costs:

* The domain with a .info extension was purchased for 182.

= Rs. 182

###### Testing and Maintenance cost:

Testing and Maintenance Cost = No. of hours \* Charges per day \* No. of Days

No. of Hours = 2 Charges per day = 80 No. of Days = 20

Testing and Maintenance Cost = 2 \* 80 \* 20

= Rs. 3,200

Overall Cost = Rs. 4000 + Rs. 45600 + Rs. 182 + Rs. 3200

**=Rs 52,982**

**USER MANUAL**

User Documentation & Easy User Manual

This chapter provides a user guide for the Blockchain-Based Voting System. It explains the different functions and components of the system, along with step-by-step instructions and screenshots to help users understand and effectively use the platform. This manual is designed to be easy to understand for all users, regardless of their technical expertise.

* + 1. Introduction to the Blockchain-Based Voting System

The Blockchain-Based Voting System is designed to provide a secure, transparent, and efficient way to conduct elections. By leveraging blockchain technology, this system ensures that every vote is recorded immutably and can be verified, enhancing the integrity and trustworthiness of the voting process. This system is designed for replacing Postal Ballot.

* + 1. User Roles and Access

The system is designed for two primary user roles:

* + - * **Voter:** Eligible citizens or members who are authorized to cast their votes in an election. Voters can register, cast their vote, and (optionally, depending on system features) verify their vote and view election results.
      * **Admin (Election Administrator/Commissioner):** Authorized personnel responsible for managing the election process. Admins can add candidates, start and stop the voting process, and view the final election results.
    1. Voter Guide: How to Use the Voting System

This section provides a step-by-step guide for voters to use the system.

Voter Registration

Before you can cast your vote, you need to register as a voter. Follow these steps:

1. **Access the Voting Platform:** Open your web browser and go to the address of the Voting System
2. **Navigate to Registration:** Click on the "Register" or "Voter Registration" button usually located on the homepage or in the navigation menu.
3. **Fill in Registration Details:** You will be presented with a registration form. Please fill in all the required details accurately. This may include:
   * Your Name
   * Voter ID or Identification Number
   * Age (Verified using **Aadhaar Offline XML Verification**])]
4. **Submit Registration:** After filling in all the details, click on the "Submit" or "Register" button at the bottom of the form.
5. **Registration Confirmation:** Upon successful registration, you will see a confirmation message on the screen. [**"Registration Successful! You can now proceed to vote once the election starts."**].

Casting Your Vote

Once registered and when the election is active, you can cast your vote. Follow these steps:

1. **Login to the System:** Return to the homepage and click on the "Login" or "Voter Login" button.
2. **Enter Login Credentials:** Enter your registered Voter ID or Identification Number and [**Specify Login Method - e.g., Password, or if using MetaMask, mention "Connect with MetaMask Wallet"**]. Click on the "Login" button.
3. **Access the Voting Dashboard:** After successful login, you will be taken to your Voter Dashboard. Here you will see the election details and the list of candidates.
4. **Select Your Candidate:** Browse through the list of candidates and select the candidate you wish to vote for. You may see candidate names, symbols, or brief descriptions.
5. **Cast Your Vote:** After selecting your candidate, click on the "Cast Vote" or "Vote Now" button.
6. **Vote Confirmation:** You will be asked to confirm your vote. Review your selection and click "Confirm Vote" to finalize your vote. [**"You will be prompted to confirm the transaction in your MetaMask wallet. Please confirm to cast your vote on the**

blockchain."]

**Vote Submission Success:** Upon successful submission, you will see a "Vote Cast Successfully" message. Your vote is now securely recorded on the blockchain.

* + 1. Admin Guide: Managing the Election Process

This section is for authorized Election Administrators to manage the voting process.

Admin Login

1. **Access Admin Panel:** Go to the admin login page.
2. **Enter Admin Credentials:** Enter your Admin Username and Password provided to you by the system administrator. Click "Login."
3. **Admin Dashboard:** After successful login, you will be taken to the Admin Dashboard. This dashboard provides access to all election management functions.

Adding Candidates

1. **Navigate to Candidates Section:** On the Admin Dashboard, find and click on the "Candidates" or "Manage Candidates" section.
2. **Add New Candidate:** Click on the "Add Candidate" or "New Candidate" button.
3. **Enter Candidate Details:** Fill in the required details for the new candidate. This may include:
   * Candidate Name
   * Party or Affiliation
   * Party Logo (Optional)
   * Constituency
4. **Save Candidate:** After entering all details, click the "Save" or "Add Candidate" button to add the candidate to the system.
5. **Candidate List Updated:** The candidate list will be updated to include the newly added candidate.

Starting the Voting Process

* + **Navigate to Election Management:** On the Admin Dashboard, find the "Election Management" or "Start/Stop Voting" section.
  + **Start Voting:** Click on the "Start Voting" or "Start Election" button.
  + **Confirmation Prompt:** You may be asked to confirm if you want to start the voting process. Confirm to proceed.
  + **Voting Started Notification:** A notification will be displayed indicating that the voting process has started. Voters can now begin casting their votes.

Stopping the Voting Process

* + 1. **Navigate to Election Management:** Go back to the "Election Management" or "Start/Stop Voting" section in the Admin Dashboard.
    2. **Stop Voting:** Click on the "Stop Voting" or "End Election" button.
    3. **Confirmation Prompt:** You will be asked to confirm if you want to stop the voting process. Confirm to proceed.
    4. **Voting Stopped Notification:** A notification will be displayed indicating that the voting process has been stopped. Voters will no longer be able to cast votes.
    5. Viewing Election Results
    6. **Navigate to Results Section:** On the Admin Dashboard, find and click on the "Results" or "View Results" section.
    7. **View Results:** The election results will be displayed. This may include:
       - Total votes cast
       - Votes per candidate (with percentages or counts)

## REFERENCES

* + - <https://ieeexplore.ieee.org/document/8457919>
    - https://archive.trufflesuite.com/ganache
    - [https://www.drawio.com](https://www.drawio.com/)
    - <https://www.youtube.com/watch?v=cjS-rAEdC_w>
    - <https://www.scirp.org/journal/paperinformation?paperid=118849>
    - <https://www.sciencedirect.com/science/article/pii/S1319157822002221>
    - https://aistudio.google.com/

## BIBLIOGRAPHY

1. Author A, Author B. "Challenges in Traditional Voting Systems." *Journal of Elections*, vol. 10, no. 2, pp. 45-58, 2020.
2. Nakamoto, S. "Bitcoin: A Peer-to-Peer Electronic Cash System."

*Bitcoin.org*, 2008. [Online]. Available: [https://bitcoin.org/bitcoin.pdf](https://www.google.com/url?sa=E&q=https%3A%2F%2Fbitcoin.org%2Fbitcoin.pdf)

1. Smith, C., and Johnson, D. "Security Issues in Electronic Voting Machines." *IEEE Security & Privacy*, vol. 15, no. 3, pp. 32-45, 2017.
2. Buterin, V. "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform." *Ethereum White Paper*, 2013. [Online].

Available: [https://ethereum.org/en/whitepaper/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fethereum.org%2Fen%2Fwhitepaper%2F)

1. "Ganache: Personal Blockchain for Ethereum Development." *Truffle Suite*. [Online]. Available: [https://trufflesuite.com/ganache/](https://www.google.com/url?sa=E&q=https%3A%2F%2Ftrufflesuite.com%2Fganache%2F)
2. "Draw.io." [Online Diagramming Tool]. Available: [https://www.drawio.com](https://www.google.com/url?sa=E&q=https%3A%2F%2Fwww.drawio.com)
3. YouTube Video: "Blockchain Voting System Explained." [Online Video]. Available: [https://www.youtube.com/watch?v=cjS-rAEdC\_w](https://www.google.com/url?sa=E&q=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DcjS-rAEdC_w) (Please replace with the actual title and creator if known from the report's context)
4. "Blockchain Technology in E-Voting Systems." *Scientific & Academic Publishing*. [Online]. Available: [https://www.scirp.org/journal/paperinformation?paperid=118849](https://www.google.com/url?sa=E&q=https%3A%2F%2Fwww.scirp.org%2Fjournal%2Fpaperinformation%3Fpaperid%3D118849) (Please replace with the actual author/title if available from the report's context)
5. "Secure and Transparent E-Voting using Blockchain." *ScienceDirect*. [Online]. Available: [https://www.sciencedirect.com/science/article/pii/S1319157822002221](https://www.google.com/url?sa=E&q=https%3A%2F%2Fwww.sciencedirect.com%2Fscience%2Farticle%2Fpii%2FS1319157822002221) (Please replace with the actual author/title if available from the report's context)
6. "Google AI Studio." [Online AI Tool]. Available: [https://aistudio.google.com/](https://www.google.com/url?sa=E&q=https%3A%2F%2Faistudio.google.com%2F)