Table of Contents

PROBLEM STATEMENT

Design and development of decentralized voting system using Blockchain.

CHAPTER 1: INTRODUCTION

1.1 Introduction to Blockchain:

Traditional voting systems, whether paper-based or electronic, often face challenges related to security, transparency, accessibility, and efficiency. Issues such as voter fraud, ballot tampering, lack of voter anonymity, and limited access to voting for remote or marginalized populations undermine the credibility and inclusivity of the electoral process. Furthermore, public trust in centralized authorities managing elections is waning due to perceived biases and vulnerabilities in the systems.

There is a critical need for a secure, transparent, decentralized, and tamper-proof voting mechanism that ensures voter privacy, eliminates fraud, and enhances accessibility. Blockchain technology, with its immutable ledger and decentralized architecture, presents an opportunity to revolutionize the voting process by addressing these challenges.

Blockchain is a distributed digital ledger technology that allows participants in a network to share and validate transactions in a secure and transparent manner without the need for intermediaries. The technology is designed to be decentralized, meaning that the data is stored on a network of computers instead of a central database. This makes it difficult to hack or manipulate the data, ensuring the integrity and security of the system.

The blockchain technology gained popularity with the emergence of Bitcoin, which was the first decentralized cryptocurrency. However, the technology has since been applied to various industries, including finance, supply chain management, healthcare, and voting, among others.

Blockchain works by creating blocks of data that are linked together in a chain, hence the name blockchain. Each block contains a unique code, known as a hash, that is generated based on the contents of the block. This hash is then used to link the block to the previous one, forming a chain of blocks.

Once a block is added to the blockchain, it cannot be altered or deleted without the consensus of the network participants. This makes the technology immutable, ensuring that the data stored on the blockchain is tamper-proof and transparent.

Overall, blockchain technology has the potential to revolutionize the way we store and share data, making it more secure, transparent, and accessible.

1.2 Decentralized Voting Using Blockchain

A decentralized voting system built on the Ethereum blockchain has the potential to revolutionize the way we conduct elections. By leveraging the security, transparency, and immutability of blockchain technology, decentralized voting systems can eliminate many of the challenges and risks associated with traditional voting systems.

In a decentralized voting system, each voter has a unique digital identity, and their vote is recorded on the blockchain, ensuring that the vote is tamper-proof and cannot be altered. Decentralized voting systems also eliminate the need for intermediaries, such as government agencies, to oversee the election process, making it more efficient and less susceptible to corruption or manipulation.

Furthermore, decentralized voting systems can increase voter participation by allowing voters to cast their ballots from anywhere in the world, as long as they have an internet connection. This can lead to a more democratic and inclusive electoral process, with greater voter engagement and higher turnout.

Overall, a decentralized voting system using the Ethereum blockchain has the potential to bring significant benefits to the electoral process, making it more secure, transparent, and accessible to everyone.

CHAPTER 2: LITERATURE SURVEY

[4] Literature survey on Online Voting System Using Blockchain

Authors: Vaibhav Anasune, Pradeep Choudhari, Madhura Kelapure, Pranali Shirke and Prasad Halgaonkar

Highly advanced security methods are necessary to introduce effective online voting system in the whole world. The aspect of security and transparency is a threat from global election with the conventional system. General elections still use a centralized system where one organization that manages it. Some of the problems that can occur in traditional electoral systems are with an organization that has full control over the database and system, it is possible to manipulate with the database. This paper presents a survey on some previous voting system that is used by different countries and organizations.

[5] A Systematic Literature Review and Meta-Analysis on Scalable Blockchain Based Electronic Voting Systems

Authors: Uzma Jafar, Mohd Juzaiddin Ab Aziz, Zarina Shukur and Hafiz Adnan Hussain

Electronic voting systems must find solutions to various issues with authentication, data privacy and integrity, transparency, and verifiability. On the other hand, Blockchain technology offers an innovative solution to many of these problems. The scalability of Blockchain has arisen as a fundamental barrier to realizing the promise of this technology, especially in electronic voting. This study seeks to highlight the solutions regarding scalable Blockchain-based electronic voting systems and the issues linked with them while also attempting to foresee future developments. A systematic literature review (SLR) was used to complete the task, leading to the selection of 76 articles in the English language from 1 January 2017 to 31 March 2022 from the famous databases. This SLR was conducted to identify well-known proposals, their implementations, verification methods, various cryptographic solutions in previous research to evaluate cost and time. It also identifies performance parameters, the primary advantages and obstacles presented by different systems, and the most common approaches for Blockchain scalability. In addition, it outlines several possible research avenues for developing a scalable electronic voting system based on Blockchain technology. This research helps future research before proposing or developing any solutions to keep in mind all the

voting requirements, merits, and demerits of the proposed solutions and provides further guidelines for scalable voting solutions.

[6] A Survey of Blockchain Based on E-voting Systems

Authors: Yousif Osman Abuidris, Rajesh Kumar and Wang Wenyong

Blockchain technology as a decentralized and distributed public ledger in a P2P network has recently gained much attention. In this technology, a linked block structure is applied, and a trusted consensus mechanism is established to synchronize data modifications, making it possible to develop a tamper-proof digital platform for data storage and sharing. We think that blockchain could be used in various interactive online systems, such as the Internet of Things, supply chain systems, voting systems, etc. The scope of this survey is to shed light on some recent contributions of the security and privacy issues associated with e-voting based on blockchain. At the end of this paper, we provided a comparison for the security and privacy requirements of the existing e-voting systems based on blockchain.

[7] Survey on Voting System using Blockchain Technology

Authors: Mayur Shirsath, Mohit Zade, Riteshkumar Talke, Praful Wake and Maya Shelke

The use of information technology has in some ways revolutionized in many sectors. E voting is said to be a symbol of modern democracy. While research on the topic is still emerging, it has mostly focused on the technical and legal issues instead of taking advantage of this technology and implementing it for good cause. Usefulness of e-voting will perform best when compared with the existing framework. The word Vote means to choose a candidate from a given list of candidates who will lead the organization or the group .The main goal of voting is to practice voting in such a way that every person votes to elect their leader. Most countries in the world, India is no exception, had trouble voting. Voting is still carried out in countries in physical mode. This physical mode process is not safe as it can be manipulated by members of voting commitment. There are many issues such as voting stations being too far and improper voting tools. The proposed flagship internet-based online voting system supported by blockchain technology solves this very problem. Blockchain technology uses encryption and hashing techniques with which it makes voting secure. In this case, each vote is considered as a unique transaction. A private blockchain is created using a peer to peer network where we store voting transactions. This application is programmed in such a way so that the details of voting are abstract from the user. Users will be given enough time for voting with the system running. The main purpose of this paper is to come up with a new unique solution, which does not require any technical

skills. Since voting is in online mode, increased voter turnout is likely. In this project, the concept of developing an electronic voting system using blockchain technology is implemented.

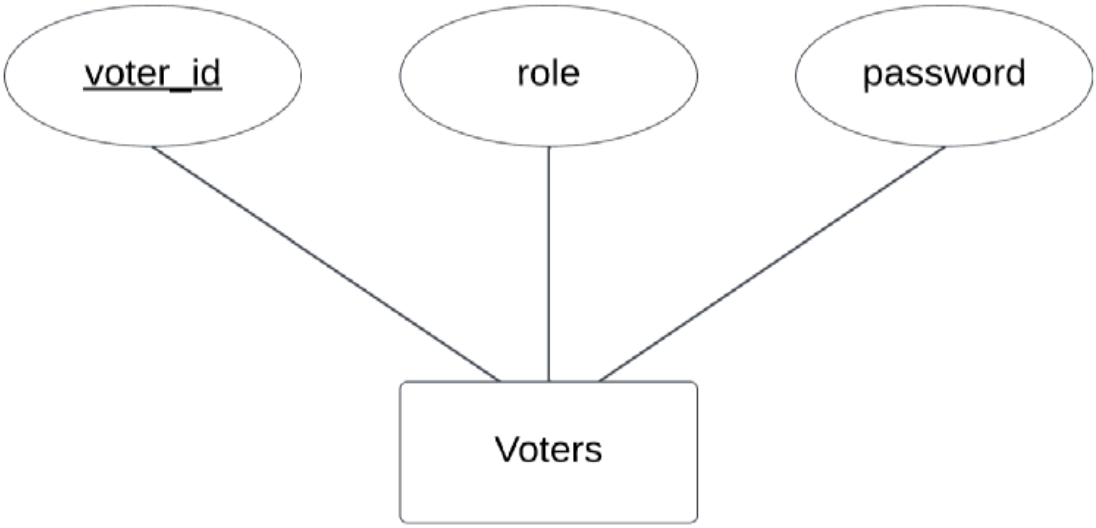
[8] A Survey on Smart Electronic Voting System Using Blockchain Technology

Authors: Naina Nagesh Dhepe and Dr. Pathan Mohd Shafi

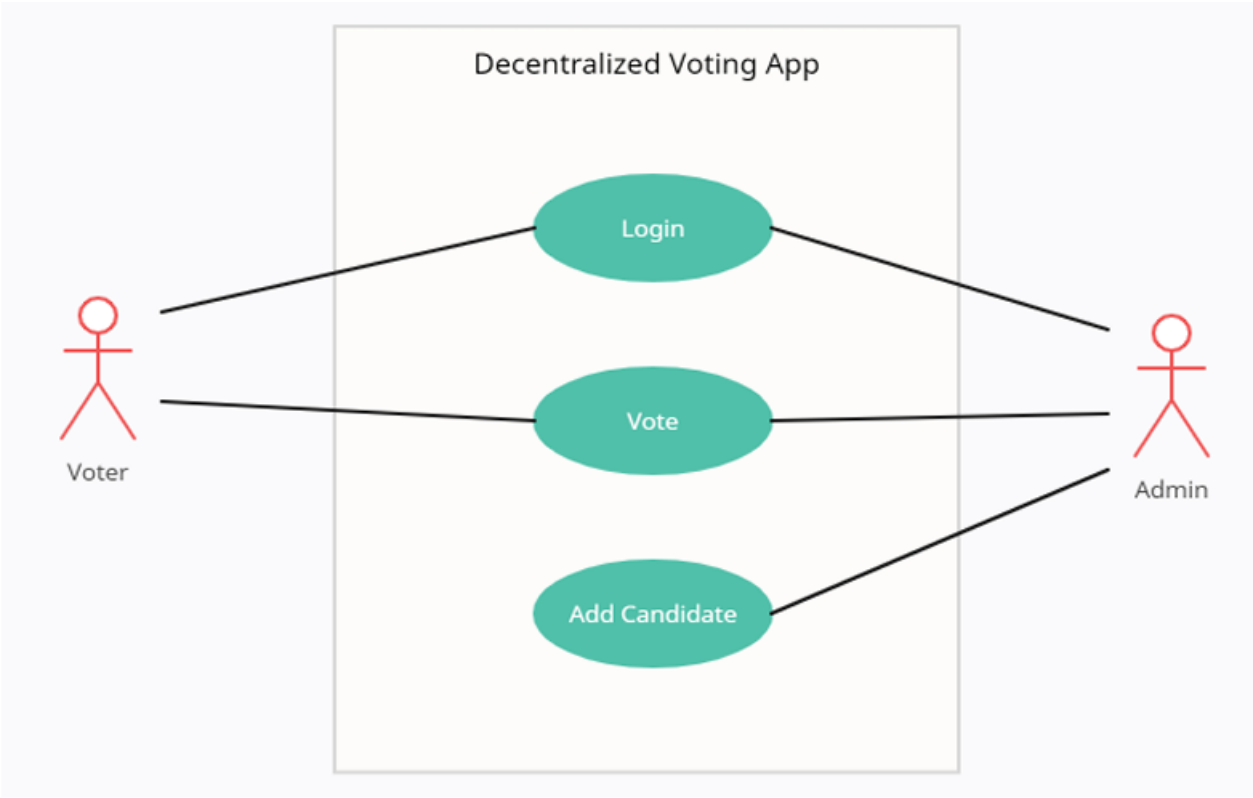
India is the world's largest democracy with a population of more than 1 billion; India has an electorate of more than 668 million and covers 543 parliamentary constituencies. Voting is the bridge between the governed and government. The last few years have brought a renewed focus on to the technology used in the voting process. The current voting system has many security holes, and it is difficult to prove even simple security properties about them. A voting system that can be proven correct has many concerns. There are some reasons for a government to use electronic systems are to increase elections activities and to reduce the elections expenses. Still there is some scope of work in electronic voting system because there is no way of identification by the electronic voting system whether the user is authentic or not and securing electronic voting machine from miscreants. The proposed system is to develop a compatible voting machine with high security by using Block-chain technology in order to increase security and transparency between the users.

CHAPTER 3: DETAILED DESIGN

ER Model:

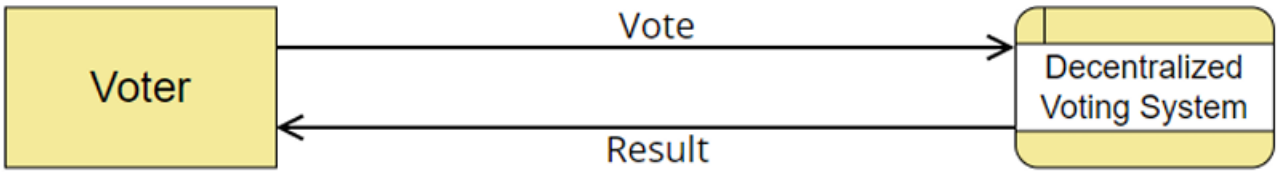


Use Case Diagram

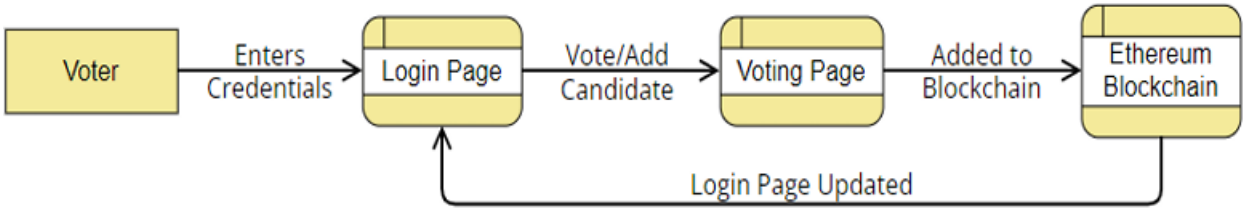


Data Flow Diagram:

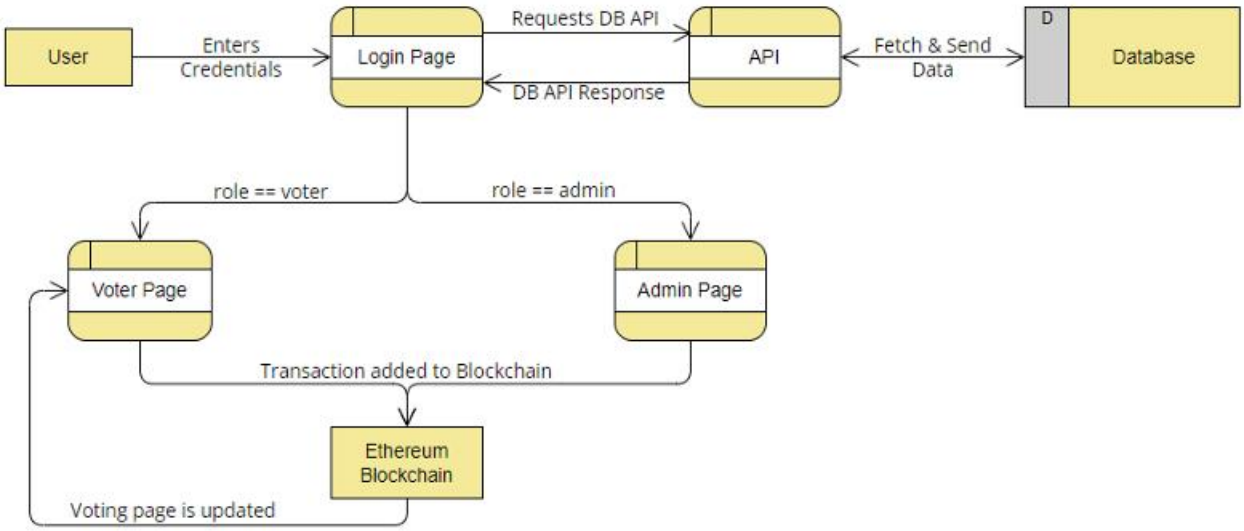
Level 0 data flow diagram:



Level 1 data flow diagram:



Level 2 data flow diagram:



CHAPTER 4: PROJECT SPECIFIC REQUIREMENTS

Requirement Analysis:

In order to effectively design and develop a system, it is important to understand and document the requirements of the system. The process of gathering and documenting the requirements of a system is known as requirement analysis. It helps to identify the goals of the system, the stakeholders and the constraints within which the system will be developed. The requirements serve as a blueprint for the development of the system and provide a reference point for testing and validation.

- **Hardware Requirements**
    - o Processor – 2 GHz or more
    - o RAM – 4 GB or more
    - o Disk Space – 100 GB or more
- **Software Requirements**
    - o Node.js (version – 18.14.0)
    - o Web3.js (version – 1.8.2)
    - o Truffle (version – 5.7.6)
    - o Solidity (version – 0.5.16)
    - o Ganache (version – 7.7.3)
    - o Metamask
    - o Python (version – 3.9)
    - o FastAPI
    - o MySQL Database (port – 3306)

CHAPTER 5: IMPLEMENTATION

CODING

1.Migrations.sol

```
pragma solidity ^0.5.15;

contract Migrations {
  address public owner;
  uint public last_completed_migration;
  modifier restricted() {
    require(msg.sender == owner, "Access restricted to owner");
    _;
  }
  constructor() public {
    owner = msg.sender;
  }
  function setCompleted(uint completed) public restricted {
    last_completed_migration = completed;
  }
  function upgrade(address new_address) public restricted {
    Migrations upgraded = Migrations(new_address);
    upgraded.setCompleted(last_completed_migration);
  }
}
```

2.Voting.sol

```
pragma solidity ^0.5.16;
contract Voting {
   struct Candidate {
      uint id;
      string name;
      string party;
      uint voteCount;
   }
   mapping (uint => Candidate) public candidates;
   mapping (address => bool) public voters;
   uint public countCandidates;
   uint256 public votingEnd;
   uint256 public votingStart;
   function addCandidate(string memory name, string memory party)
   public  returns(uint) {
            countCandidates ++;
            candidates[countCandidates] = Candidate(countCandidates, name, party, 0);
            return countCandidates;
   }
   function vote(uint candidateID) public {
     require((votingStart <= now) && (votingEnd > now));
     require(candidateID > 0 && candidateID <= countCandidates);
     //daha önce oy kullanmamıs olmalı
     require(!voters[msg.sender]);
     voters[msg.sender] = true;
     candidates[candidateID].voteCount ++;
  }
function checkVote() public view returns(bool){
      return voters[msg.sender];
   }
function getCountCandidates() public view returns(uint) {
      return countCandidates;
   }
function getCandidate(uint candidateID) public view returns (uint,string memory, string
```

```
memory,uint) {
return(candidateID,candidates[candidateID].name,candidates[candidateID].party,candidates[candidateID].voteCount);
  }
  function setDates(uint256 _startDate, uint256 _endDate) public{
    require((votingEnd == 0) && (votingStart == 0) && (_startDate + 1000000 > now) && (_endDate > _startDate));
    votingEnd = _endDate;
    votingStart = _startDate;
  }
  function getDates() public view returns (uint256,uint256) {
   return (votingStart,votingEnd);
  }
}
```

3.app.js

```
const Web3 = require('web3');
const contract = require('@truffle/contract');
const votingArtifacts = require('../../build/contracts/Voting.json');
var VotingContract = contract(votingArtifacts);
window.App = {
 eventStart: async function () {
  try {
   // Request accounts and set provider
   const accounts = await ethereum.request({ method: 'eth_requestAccounts' });
   const account = accounts[0];
   VotingContract.setProvider(window.ethereum);
   VotingContract.defaults({ from: account, gas: 6654755 });
   // Display the connected account
   $("#accountAddress").html("Your Account: " + account);
   const instance = await VotingContract.deployed();
   // Fetch the total number of candidates
   const countCandidates = await instance.getCountCandidates();
   // Fetch candidates in parallel using Promise.all
```

```
    const candidatePromises = [];
    for (let i = 0; i < countCandidates; i++) {
      candidatePromises.push(instance.getCandidate(i + 1));
    }
    const candidates = await Promise.all(candidatePromises);
    // Append candidates to the UI
    candidates.forEach(data => {
      const viewCandidates = `<tr>
        <td><input class="form-check-input" type="radio" name="candidate" value="${data[0]}" id="${data[0]}"></td>
        <td>${data[1]}</td>
        <td>${data[2]}</td>
        <td>${data[3]}</td>
      </tr>`;
      $("#boxCandidate").append(viewCandidates);
    });
    // Add event listener for adding a candidate
    $('#addCandidate').off('click').on('click', async function () {
      const nameCandidate = $('#name').val();
      const partyCandidate = $('#party').val();
      if (!nameCandidate || !partyCandidate) {
        alert("Please fill in both name and party fields.");
        return;
      }
      try {
        await instance.addCandidate(nameCandidate, partyCandidate);
        alert("Candidate added successfully!");
        window.location.reload();
      } catch (err) {
        console.error("Error adding candidate:", err);
        alert("Failed to add candidate. Please try again.");
      }
    });
    // Add event listener for setting dates
```

```
    $('#addDate').off('click').on('click', async function () {
      const startDate = Date.parse($("#startDate").val()) / 1000;
      const endDate = Date.parse($("#endDate").val()) / 1000;
      if (isNaN(startDate) || isNaN(endDate)) {
        alert("Please select valid start and end dates.");
        return;
      }
      try {
        await instance.setDates(startDate, endDate);
        alert("Dates updated successfully!");
        window.location.reload();
      } catch (err) {
        console.error("Error setting dates:", err);
        alert("Failed to set dates.");
      }
    });

    // Display voting dates
    const dates = await instance.getDates();
    const startDate = new Date(dates[0] * 1000).toDateString();
    const endDate = new Date(dates[1] * 1000).toDateString();
    $("#dates").text(${startDate} - ${endDate});
    // Check if the user has voted
    const hasVoted = await instance.checkVote();
    if (!hasVoted) {
      $("#voteButton").attr("disabled", false);
    }
  } catch (err) {
    console.error("Error initializing app:", err);
  }
},
 vote: async function () {
  const candidateID = $("input[name='candidate']:checked").val();
  if (!candidateID) {
```

```
    $("#msg").html("<p>Please vote for a candidate.</p>");
    return;
   }
   try {
    const instance = await VotingContract.deployed();
    await instance.vote(parseInt(candidateID));
    $("#voteButton").attr("disabled", true);
    $("#msg").html("<p>Voted</p>");
    window.location.reload();
   } catch (err) {
    console.error("Error during voting:", err);
    alert("Failed to cast vote. Please try again.");
   }
  }
};
window.addEventListener("load", function () {
  if (typeof window.ethereum !== "undefined") {
   console.warn("Using web3 detected from external source like Metamask");
   window.web3 = new Web3(window.ethereum);
  } else {
   console.warn("No web3 detected. Please use Metamask.");
   window.web3 = new Web3(new Web3.providers.HttpProvider("http://127.0.0.1:9545"));
  }
  window.App.eventStart();
});
```

4.login.js

```
const loginForm = document.getElementById('loginForm');
loginForm.addEventListener('submit', (event) => {
  event.preventDefault();
  const voter_id = document.getElementById('voter-id').value;
  const password = document.getElementById('password').value;
  const token = voter_id;
  const headers = {
```

```
  'method': "GET",
  'Authorization': Bearer ${token},
 };
 fetch(http://127.0.0.1:8000/login?voter_id=${voter_id}&password=${password}, { headers })
 .then(response => {
  if (response.ok) {
   return response.json();
  } else {
   throw new Error('Login failed');
  }
 })
 .then(data => {
  if (data.role === 'admin') {
   console.log(data.role)
   localStorage.setItem('jwtTokenAdmin', data.token);
   window.location.replace(http://127.0.0.1:8080/admin.html?Authorization=Bearer
${localStorage.getItem('jwtTokenAdmin')});
  } else if (data.role === 'user'){
   localStorage.setItem('jwtTokenVoter', data.token);
   window.location.replace(http://127.0.0.1:8080/index.html?Authorization=Bearer
${localStorage.getItem('jwtTokenVoter')});
  }
 })
 .catch(error => {
  console.error('Login failed:', error.message);
 });
});
```

5. main.py

```
# Import required modules
import dotenv
import os
import mysql.connector
from fastapi import FastAPI, HTTPException, status, Request
```

```
from fastapi.middleware.cors import CORSMiddleware
from fastapi.encoders import jsonable_encoder
from mysql.connector import errorcode
import jwt
# Loading the environment variables
dotenv.load_dotenv()
# Initialize the todoapi app
app = FastAPI()
# Define the allowed origins for CORS
origins = [
   "http://localhost:8080",
   "http://127.0.0.1:8080",
]
# Add CORS middleware
app.add_middleware(
   CORSMiddleware,
   allow_origins=origins,
   allow_credentials=True,
   allow_methods=["*"],
   allow_headers=["*"],
)
# Connect to the MySQL database
try:
   cnx = mysql.connector.connect(
      user=os.environ['MYSQL_USER'],
      password=os.environ['MYSQL_PASSWORD'],
      host=os.environ['MYSQL_HOST'],
      database=os.environ['MYSQL_DB'],
   )
   cursor = cnx.cursor()
except mysql.connector.Error as err:
   if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
      print("Something is wrong with your user name or password")
   elif err.errno == errorcode.ER_BAD_DB_ERROR:
```

```
        print("Database does not exist")
    else:
        print(err)
# Define the authentication middleware
async def authenticate(request: Request):
    try:
        api_key = request.headers.get('authorization').replace("Bearer ", "")
        cursor.execute("SELECT * FROM voters WHERE voter_id = %s", (api_key,))
        if api_key not in [row[0] for row in cursor.fetchall()]:
            raise HTTPException(
                status_code=status.HTTP_401_UNAUTHORIZED,
                detail="Forbidden"
            )
    except:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Forbidden"
        )

# Define the POST endpoint for login
@app.get("/login")
async def login(request: Request, voter_id: str, password: str):
    await authenticate(request)
    role = await get_role(voter_id, password)
    # Assuming authentication is successful, generate a token
    token = jwt.encode({'password':password,'voter_id':voter_id,'role': role}, os.environ['SECRET_KEY'],
algorithm='HS256')
    return {'token': token, 'role': role}
# Replace 'admin' with the actual role based on authentication
async def get_role(voter_id, password):
    try:
        cursor.execute("SELECT role FROM voters WHERE voter_id = %s AND password = %s", (voter_id,
password,))
        role = cursor.fetchone()
```

```
    if role:
      return role[0]
    else:
      raise HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="Invalid voter id or password"
      )
  except mysql.connector.Error as err:
    print(err)
    raise HTTPException(
      status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
      detail="Database error"
    )
```

6. package.json

```
{
 "name": "decentralized-voting-system-main",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
  "test": "echo \"Error: no test specified\" && exit 1"
 },
 "author": "",
 "license": "ISC",
 "dependencies": {
  "jsonwebtoken": "^9.0.0",
  "@truffle/contract": "^4.6.18",
  "browserify": "^17.0.0",
  "dotenv": "^16.0.3",
  "express": "^4.18.2",
  "web3": "^1.9.0"
 }
}
```

TESTING

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. It includes a set of techniques and methods to identify defects, bugs, performance issues and providing a reliable and quality product. The goal is to identify issues as early as possible and improve the overall quality of the system.

Types of Testing

Unit Testing

Unit testing is a type of testing that is used to evaluate the individual units or components of a software system. This type of testing helps ensure that each unit or component of the system is working correctly and is able to perform its intended function.

Integration Testing

Integration testing is a type of testing that is used to evaluate how well the different units or components of a software system work together. This type of testing helps to identify and resolve issues related to compatibility, performance, and data flow between the different units or components.

Functional Testing

Functional testing is a type of testing that is used to evaluate how well a system or software performs the specific functions or tasks that it is designed to perform. It is done by testing the system or software with various inputs and verifying that the outputs are correct. This type of testing ensures that the system or software is working as intended and is able to perform the functions it was designed to perform.

White Box Testing

White box testing, also known as structural testing or glass-box testing, is a type of testing that examines the internal structure and implementation of a software system. It involves testing the code itself and checking that it is functioning correctly and adhering to coding standards. This type of testing helps to identify and resolve issues related to logic, control flow, and data structures within the system.

Black Box Testing

Black box testing, also known as functional testing, is a type of testing that examines the external behavior and interfaces of a software system. It involves testing the system from the user's

perspective, without looking at the internal structure or implementation, and checking that it is functioning correctly and meeting the requirements. This type of testing helps to identify and resolve issues related to usability, compatibility, and performance.

Test Results

Test Case 1

Test Case No.	1
Test Type	Unit Test
Name of Test	Checking JWT Authorization
Test Case Description	The objective of this test case is to check jwt authorization.
Input	Login and Password
Expected Output	User should not be able to login without proper authorization.
Actual Output	User cannot access voting or admin page without authorization.
Result	Pass
Comments	Working properly.

Test Case 2

Test Case No.	2
Test Type	Functional Test
Name of Test	Verify user login
Test Case Description	The objective of this test case is to verify that user can login to the voting portal.
Input	Voter_id and password
Expected Output	User must be able to login if credentials match the database, else unauthorized error is shown.
Actual Output	User is able to login with correct credentials only.
Result	Pass
Comments	Working properly.

Test Case 3

Test Case No.	3
Test Type	Unit Test
Name of Test	Verify candidate registration
Test Case Description	The objective of this test case is to verify that candidate can be registered by admin.
Input	Candidate name and party.
Expected Output	Registration transaction should be successful.
Actual Output	Registration transaction is successful.
Result	Pass
Comments	Working properly.

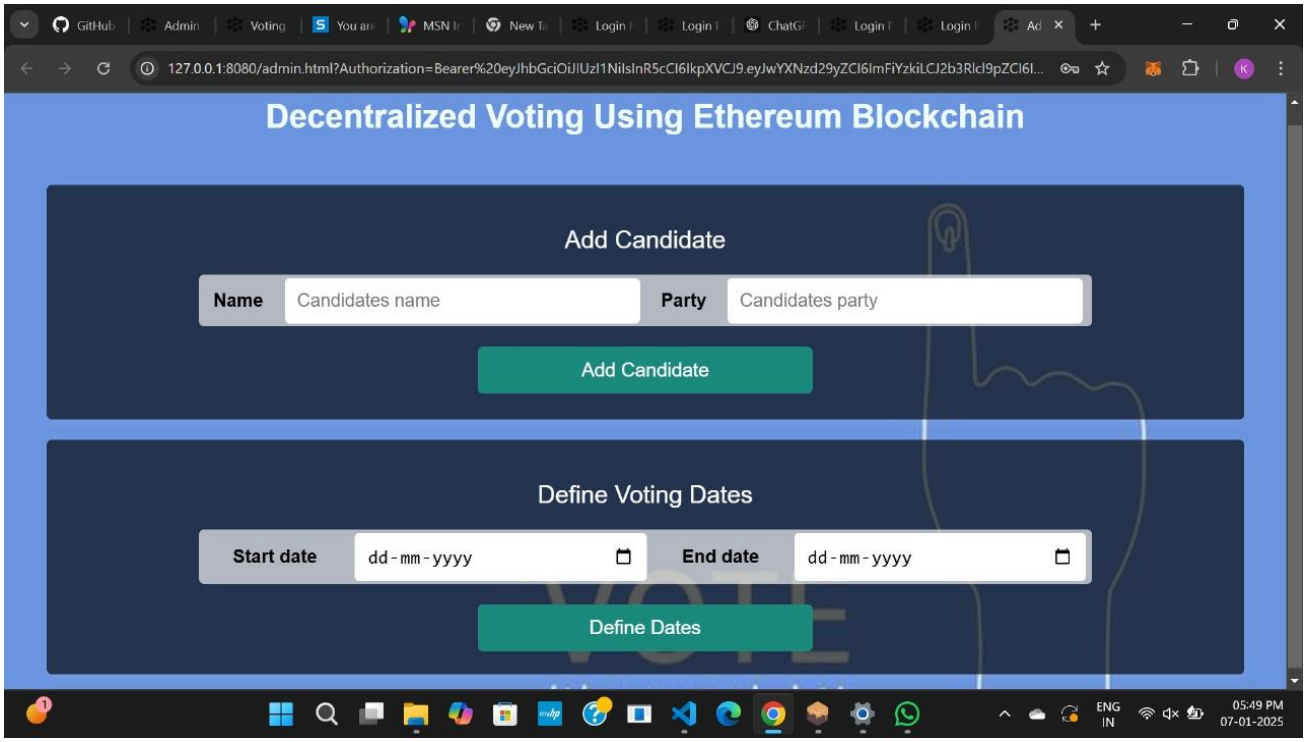
Test Case 4

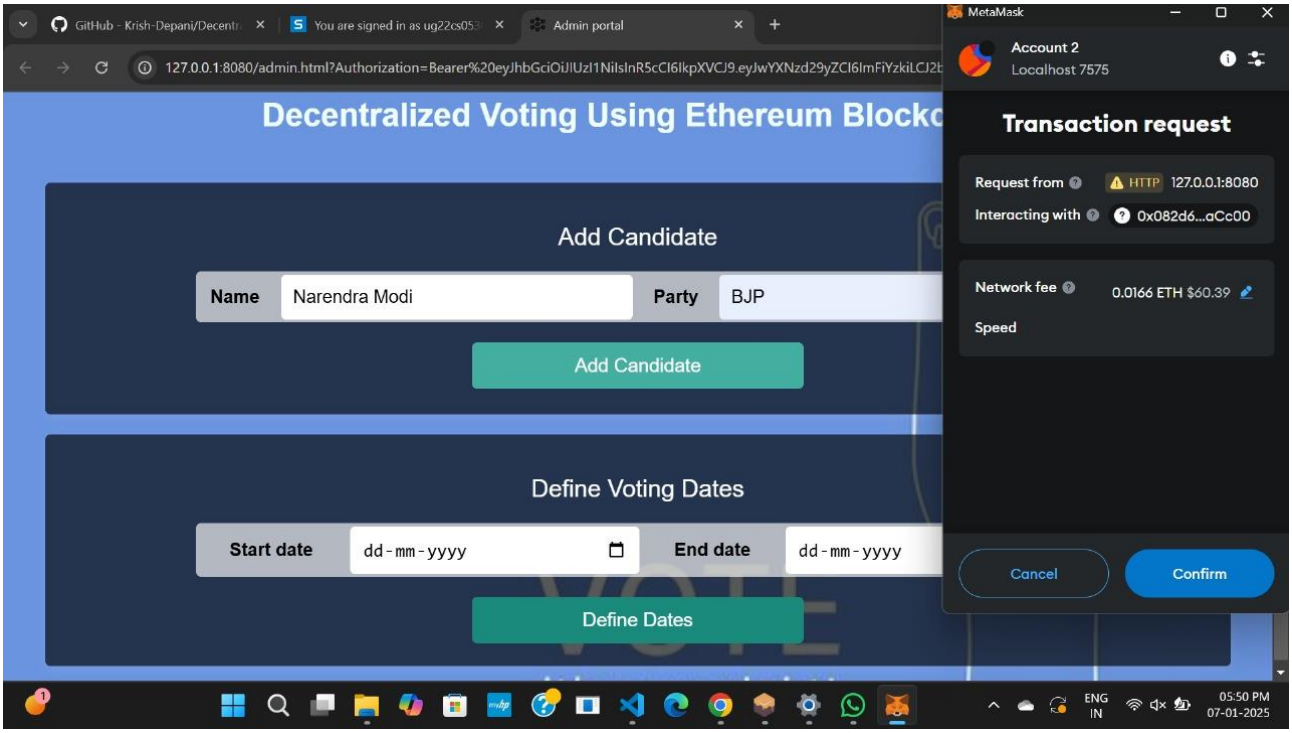
Test Case No.	4
Test Type	Unit Test
Name of Test	Verify date registration
Test Case Description	The objective of this test case is to verify that date of voting can be specified by admin.
Input	Starting and ending date
Expected Output	Date transaction should be successful.
Actual Output	Date transaction is successful.
Result	Pass
Comments	Working properly.

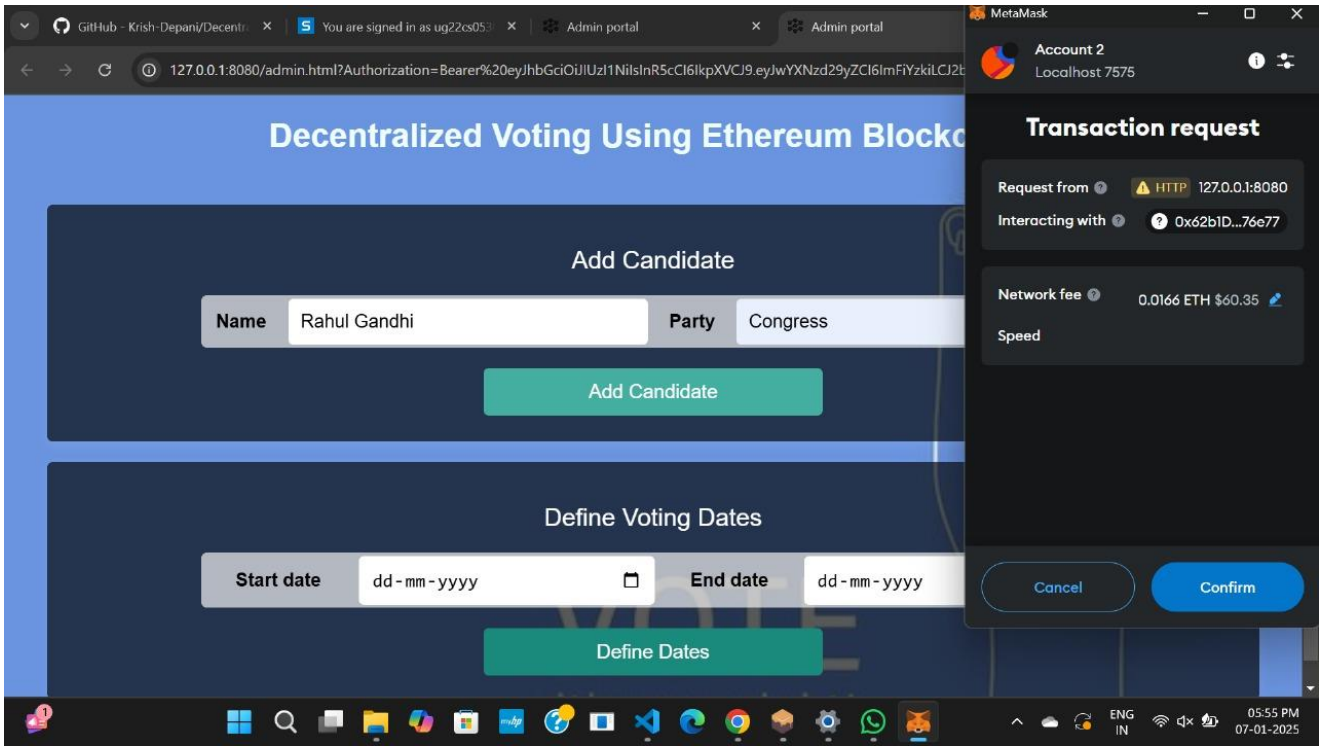
Test Case 5

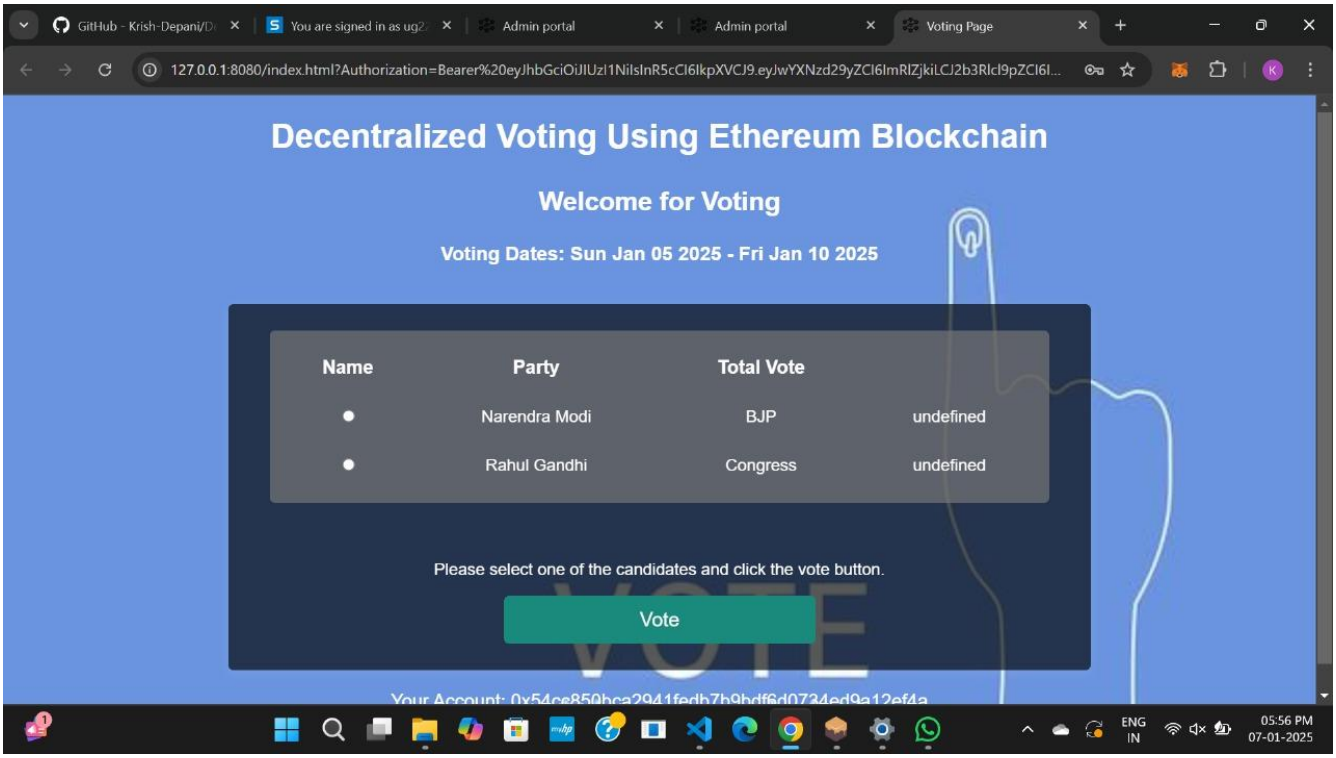
Test Case No.	5
Test Type	Functional Test
Name of Test	Verify voting
Test Case Description	The objective of this test case is to verify that voter is able to cast their vote.
Input	Select a candidate and click “Vote” button.
Expected Output	Vote transaction should be successful.
Actual Output	Vote transaction is successful.
Result	Pass
Comments	Working properly.

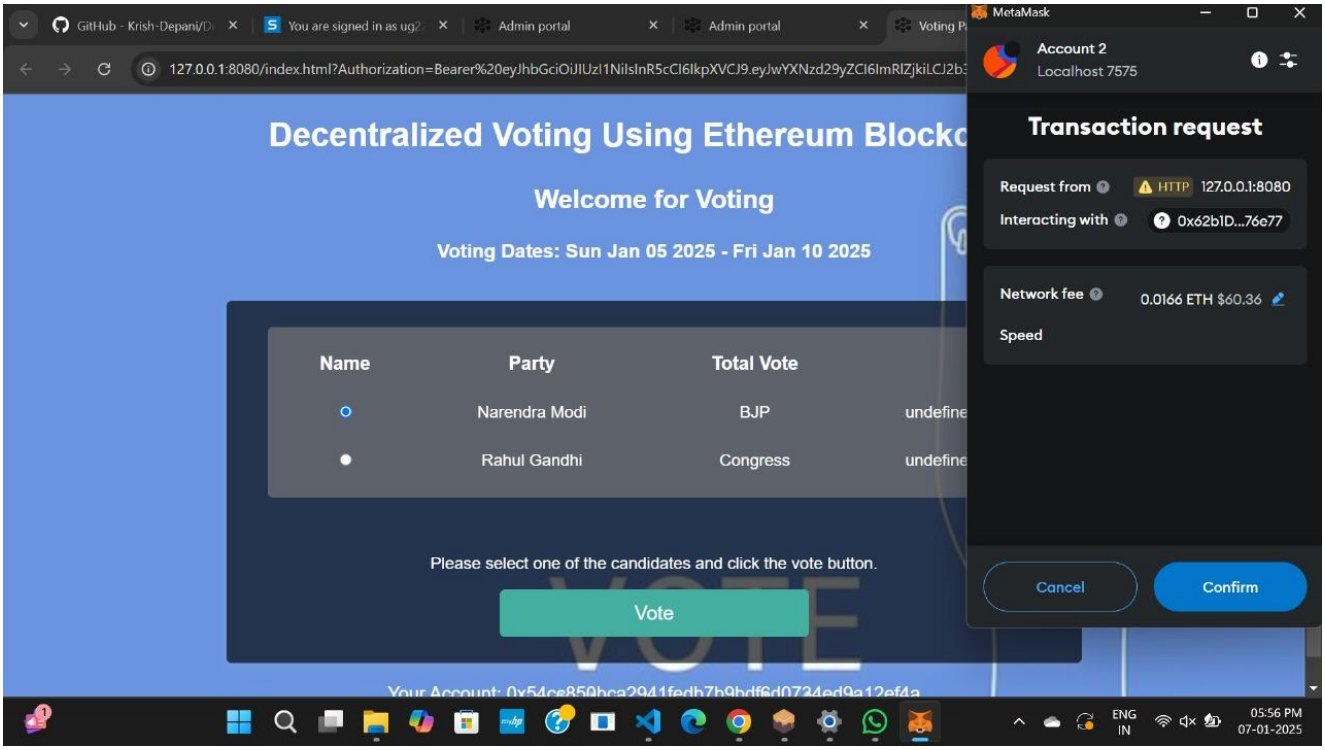
CHAPTER 6: RESULTS




Admin portal
Decentralized Voting Using Ethereum Blockc
Add Candidate
Name
Narendra Modi
Party
BJP
Add Candidate
Define Voting Dates
Start date
dd-mm-yyyy
End date
dd-mm-yyyy
Define Dates
MetaMask
Account 2
Localhost 7575
Transaction request
Request from
HTTP 127.0.0.1:8080
Interacting with
0x082d6...aCc00
Network fee
0.0166 ETH $60.39
Speed
Cancel
Confirm
ENG
IN
05:50 PM
07-01-2025


Admin portal
Admin portal
Decentralized Voting Using Ethereum Blockc
Add Candidate
Name
Rahul Gandhi
Party
Congress
Add Candidate
Define Voting Dates
Start date
dd-mm-yyyy
End date
dd-mm-yyyy
Define Dates
MetaMask
Account 2
Localhost 7575
Transaction request
Request from
HTTP 127.0.0.1:8080
Interacting with
0x62b1D...76e77
Network fee
0.0166 ETH $60.35
Speed
Cancel
Confirm
ENG
IN
05:55 PM
07-01-2025


Decentralized Voting Using Ethereum Blockchain
Welcome for Voting
Voting Dates: Sun Jan 05 2025 - Fri Jan 10 2025
Name
Party
Total Vote
Narendra Modi
BJP
undefined
Rahul Gandhi
Congress
undefined
Please select one of the candidates and click the vote button.
Vote


MetaMask
Account 2
Localhost 7575
Transaction request
Request from
HTTP 127.0.0.1:8080
Interacting with
0x62b1D...76e77
Network fee
0.0166 ETH $60.36
Speed
Cancel
Confirm
Decentralized Voting Using Ethereum Blockc
Welcome for Voting
Voting Dates: Sun Jan 05 2025 - Fri Jan 10 2025
Name
Party
Total Vote
Narendra Modi
BJP
Rahul Gandhi
Congress
Please select one of the candidates and click the vote button.
Vote

CHAPTER 7: CONCLUSION and FUTURE SCOPE

Conclusion:

Decentralized Voting with Ethereum Blockchain offers a robust and transparent solution for secure elections. By leveraging blockchain technology, it ensures the integrity of votes and provides a tamper-proof platform. With continued enhancements, including improved user experience, scalability, and integration with other cutting-edge technologies, it has the potential to revolutionize the democratic process and empower citizens to participate in a trusted and efficient voting system. It represents a significant step towards building a more democratic and accountable society.

Future Scope:

In future iterations, the decentralized voting system can be enhanced by implementing additional features such as real-time vote counting, secure voter identification mechanisms, advanced data analytics for voter insights, and integration with emerging technologies like artificial intelligence and biometrics. These enhancements will further enhance the efficiency, security, and accessibility of the voting process, making it more inclusive and trustworthy.

Incorporating advanced technologies such as iris sensing for voter authentication can revolutionize the security and reliability of voter identification. Iris recognition, with its high accuracy and resistance to spoofing, ensures that each individual is uniquely identified, eliminating the risk of voter fraud.

The decentralized voting system can also be designed to assist local authorities by providing tools for seamless election management, including automated voter roll updates, efficient polling station setup, and real-time monitoring of election activities. These features can empower election officials to address logistical challenges effectively and ensure smooth election operations.

A decentralized voting system can streamline elections for college and school events like class representative (CR) or school parliament elections. Students can securely cast their votes using their smartphones or biometric authentication, ensuring transparency and fairness. Real-time vote

counting and instant results make the process efficient and engaging, fostering a practical understanding of democratic principles among students.

By leveraging these innovations, the system can facilitate elections even in remote or underserved regions, promoting greater participation and ensuring the democratic process is accessible to all citizens.

REFERENCES

[1] E-Voting System based on Blockchain Technology, Department of Computer Science & Engineering and Information Technology Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India).

[2] Blockchain Technology and Applications, Written by Pethuru Raj, Ganesh Chandra Deka, and S. Raj Pethuru.

[3] Github Repositry: https://github.com/mehtaAnsh/BlockChainVoting.

[4] Literature survey on Online Voting System Using Blockchain, Vaibhav Anasune, Pradeep Choudhari, Madhura Kelapure, Pranali Shirke and Prasad Halgaonkar.

[5] A Systematic Literature Review and Meta-Analysis on Scalable Blockchain Based Electronic Voting Systems, Uzma Jafar, Mohd Juzaiddin Ab Aziz, Zarina Shukur and Hafiz Adnan Hussain.

[6] A Survey of Blockchain Based on E-voting Systems, Yousif Osman Abuidris, Rajesh Kumar and Wang Wenyong

[7] Survey on Voting System using Blockchain Technology, Mayur Shirsath, Mohit Zade, Riteshkumar Talke, Praful Wake and Maya Shelke.

[8] A Survey on Smart Electronic Voting System Using Blockchain Technology, Naina Nagesh Dhepe and Dr. Pathan Mohd Shafi