BLOCKCHAIN TECHNOLOGY
CS-411

SEMESTER PROJECT
BLOCKCHAIN BASED DECENTRALIZED VOTING SYSTEM

(ASSIGNMENT # 4)
LIVE DEMOSTRATION
& WORKING OF
PROTOTYPE

NAME: ALI NAUFIL
REG #: 2021083
SUPERVISOR: M.AHMED NAWAZ

LIVE DEMOSTARATION AND WORKING YOUTUBE LINK:

https://www.youtube.com/watch?v=vZk9leWHi_U

GITHUB REPOSITORY LINK:

https://github.com/Ali-Naufil-09/Decentralized-Voting-System-Using-Ethereum-Blockchain

INTRODUCTION

1.1 Introduction to Blockchain

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.

RESEARCH

2.1 Survey on Online Voting System Using Blockchain

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 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.

2.2 A Systematic Review and Meta-Analysis on Scalable Blockchain- Based Electronic Voting Systems

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. 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.

2.3 A Survey of Blockchain Based on E-voting Systems

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.

2.4 A Survey on Smart Electronic Voting System Using Blockchain Technology

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 the 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.

EXISTING SYSTEM

3.1 Brief Explanation of existing system

The existing voting system typically involves voters physically visiting a designated polling place to cast their vote on paper ballots. These ballots are then manually counted and recorded. Some countries also have electronic voting systems in place, which allow voters to cast their votes electronically through machines or the internet. However, electronic voting systems have faced criticism due to security concerns and potential vulnerabilities.

3.2 Disadvantages of existing system

1. **Lack of transparency**: In most voting systems, it's difficult for voters to know whether their vote was counted correctly, and for observers to ensure that the vote counting process is fair.
2. **Vulnerability to fraud**: Both paper ballots and electronic voting machines can be vulnerable to tampering, hacking and other types of fraud. This can be especially problematic when there is no paper trail or other way to audit the results.
3. **Slow results**: Counting paper ballots can be a time-consuming and labor-intensive process, which can delay the announcement of election results.
4. **Cost**: Running a traditional voting system can be expensive, requiring the hiring of poll workers, the purchase of voting machines or paper ballots, and the rental of polling places.
5. **Centralization**: Many traditional voting systems are centralized, meaning that they are controlled by a small number of authorities. This can create the potential for abuse of power or manipulation of the voting process.
6. **Limited Accessibility**: Some voting systems require voters to travel to specific polling places, which can be difficult or impossible for people with disabilities, limited mobility, or other challenges. This can result in voter disenfranchisement.

PROPOSED SYSTEM

4.1 Brief explanation of proposed system

The proposed decentralized voting system using Ethereum blockchain aims to provide a transparent and tamper-proof solution for conducting elections. By leveraging smart contracts on the Ethereum network, the system enables secure and anonymous voting, while ensuring the integrity and immutability of the voting data. This would increase voter trust in the election process and reduce the risk of fraud or manipulation.

4.2 Advantages of Proposed System

- Decentralization ensures that no party controls the voting process.
- Transparency throughout the voting process.
- It is tamper proof.
- Voters can vote from any part of the world.
- This method of voting is cost effective.
- The results are provided in real time.

4.3 Objectives of the Proposed Research

1. **Security:** The proposed system aims to provide a secure platform for conducting elections, eliminating the possibility of tampering with votes, and ensuring that the election results are transparent and verifiable.

2. **Transparency:** The proposed system aims to provide complete transparency to the voters, allowing them to view the entire voting process, including the vote counting and results.

3. **Accessibility:** The proposed system aims to make the voting process more accessible to all eligible voters by eliminating the need for physical presence at a polling station, thus increasing voter turnout.

4. **Efficiency:** The system aims to increase the efficiency of the voting process by reducing the time and resources required to conduct elections. Since the system is automated and eliminates the need for intermediaries, it can significantly reduce the cost and time associated with traditional voting methods.

5. **Trust:** The proposed system aims to increase trust in the voting process by providing a transparent and tamper-proof mechanism for recording and tallying votes.

4.4 System Architecture

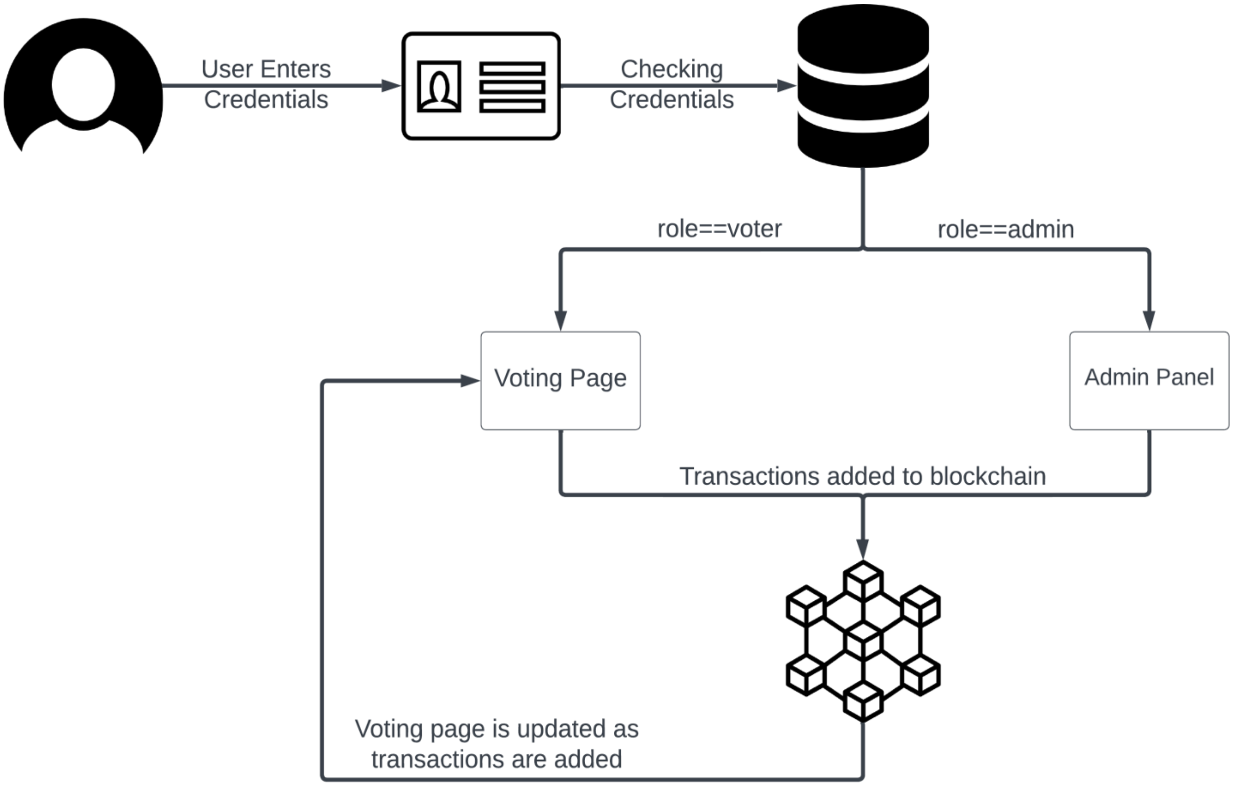

Figure 1 System Architecture

User enters the credentials (voter id & password) and they are matched with the database. If the match is found user is either redirected to admin page or voter page as per their role corresponding to the credentials in the database. Once the admin is logged in he/she can start the voting process by adding candidates and defining dates. Voter can vote once the voting process has been started. Once the voter has voted the transaction is recorded to the blockchain and the voting page is updated with real-time votes.

MODULES

5.1 Modules

1. **Voter -** The voter module is designed for individuals who are eligible to participate in the voting process. It provides functionalities related to the voting experience and ensures the integrity and security of the votes. The main features of the voter module include:

    a. Voters can securely authenticate themselves to access the voting system using their unique credentials.

    b. Voters can access information about the candidates running for various positions, such as their names, parties, and other relevant details.

    c. Voters can verify the status of their votes and ensure that their choices are accurately recorded in the blockchain.

2. **Admin -** The admin module is designed for administrators or election officials responsible for managing and overseeing the voting system. It provides functionalities to configure and monitor the voting process. The main features of the admin module include:

    a. Admins can set up the system parameters, such as defining the start and end dates of the voting period, candidate registration, and other administrative settings.

    b. Admin can manually verify the candidate and can start the voting process.

SYSTEM DESIGN

6.1 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**
  - Processor – 2 GHz or more
  - RAM – 4 GB or more
  - Disk Space – 100 GB or more

- **Software Requirements**
  - Node.js
  - Web3.js
  - Truffle
  - Solidity
  - Ganache
  - MetaMask
  - Python
  - MySQL Database

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 ^0.5.15;

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: function() {
    window.ethereum.request({ method: 'eth_requestAccounts' });
    VotingContract.setProvider(window.ethereum)
    VotingContract.defaults({from:
window.ethereum.selectedAddress,gas:6654755})

    // Load account data
    App.account = window.ethereum.selectedAddress;
    $("#accountAddress").html("Your Account: " +
window.ethereum.selectedAddress);
    VotingContract.deployed().then(function(instance){
     instance.getCountCandidates().then(function(countCandidates){

            $(document).ready(function(){
              $('#addCandidate').click(function() {
                  var nameCandidate = $('#name').val();
                  var partyCandidate = $('#party').val();
                 instance.addCandidate(nameCandidate,partyCandidate).the
n(function(result){ })

            });
              $('#addDate').click(function(){
                  var startDate =
Date.parse(document.getElementById("startDate").value)/1000;

                  var endDate
=  Date.parse(document.getElementById("endDate").value)/1000;

                  instance.setDates(startDate,endDate).then(function(rsl
t){

                    console.log("tarihler verildi");
                  });

              });

                instance.getDates().then(function(result){
```

```
                var startDate = new Date(result[0]*1000);
                var endDate = new Date(result[1]*1000);

                $("#dates").text(
startDate.toDateString(("#DD#/#MM#/#YYYY#")) + " - " +
endDate.toDateString("#DD#/#MM#/#YYYY#"));
              }).catch(function(err){
                console.error("ERROR! " + err.message)
              });
          });

          for (var i = 0; i < countCandidates; i++ ){
            instance.getCandidate(i+1).then(function(data){
              var id = data[0];
              var name = data[1];
              var party = data[2];
              var voteCount = data[3];
              var viewCandidates = `<tr><td> <input class="form-check-
input" type="radio" name="candidate" value="${id}" id=${id}>` + name +
"</td><td>" + party + "</td><td>" + voteCount + "</td></tr>"
              $("#boxCandidate").append(viewCandidates)
            })
        }

        window.countCandidates = countCandidates
      });

      instance.checkVote().then(function (voted) {
          console.log(voted);
          if(!voted)  {
            $("#voteButton").attr("disabled", false);

          }
      });

    }).catch(function(err){
      console.error("ERROR! " + err.message)
    })
  },

  vote: function() {
    var candidateID = $("input[name='candidate']:checked").val();
    if (!candidateID) {
      $("#msg").html("<p>Please vote for a candidate.</p>")
      return
    }
    VotingContract.deployed().then(function(instance){
      instance.vote(parseInt(candidateID)).then(function(result){
        $("#voteButton").attr("disabled", true);
        $("#msg").html("<p>Voted</p>");
         window.location.reload(1);
      })
    }).catch(function(err){
      console.error("ERROR! " + err.message)
    })
  }
}

window.addEventListener("load", function() {
  if (typeof web3 !== "undefined") {
    console.warn("Using web3 detected from external source like
Metamask")
    window.eth = new Web3(window.ethereum)
  } else {
    console.warn("No web3 detected. Falling back to
http://localhost:9545. You should remove this fallback when you deploy
live, as it's inherently insecure. Consider switching to Metamask for
deployment. More info here:
http://truffleframework.com/tutorials/truffle-and-metamask")
    window.eth = 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

```python
# 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",
  "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.

8.1 Types of Testing

8.1.1 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.

8.1.2 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.

8.1.3 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.

8.2 Test Results

8.2.1 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.

8.2.2 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.

8.2.3 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.

8.2.4 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.

8.2.5 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.

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.