PROJECT REPORT ON

**DECENTRALIZED VOTING SYSTEM USING BLOCKCHAIN**

Submitted to Department of Computer Applications

in partial fulfillment for the award of the degree of

BACHELOR OF COMPUTER APPLICTIONS

**Batch (2023-2026)**

***Submitted by*** Kiran Chand GE-23213564

**Under the Guidance of Ms. Vandana Rawat**



GRAPHIC ERA DEEMED TO BE UNIVERSITY DEHRADUN

December- 2024



**CANDIDATE’S DECLARATION**

I hereby certify that the work presented in this project report entitled *“DECENTRALIZED VOTING SYSTEM USING BLOCKCHAIN”* in partial fulfilment of the requirements for the award of the degree of Bachelor of Computer Applications is a bonafide work carried out by me during the period of July 2024 to December 2024 under the supervision of Ms. Vandana Rawat , Department of Computer Application, Graphic Era Deemed to be University, Dehradun, India.

This work has not been submitted elsewhere for the award of a degree/diploma/certificate.

## Name and Signature of Candidate

This is to certify that the above mentioned statement in the candidate’s declaration is correct to the best of my knowledge.

**Date: Name and Signature of Guide**

**Signature of Supervisor Signature of External Examiner**

# HOD

**CERTIFICATE OF ORIGINALITY**

This is to certify that the project report entitled DECENTRALIZED VOTING SYSTEM USING BLOCKCHAIN submitted to **Graphic Era University, Dehradun** in partial fulfilment of the requirement for the award of the degree of **BACHELOR OF COMPUTER APPLICATION,** is an authentic and original work carried out by Ms. Kiran Chand with enrolment number 2103564 under my supervision and guidance.

The matter embodied in this project is genuine work done by the student and has not been submitted whether to this University or to any other University / Institute for the fulfilment of the requirements of any course of study.

…………………………. ………………………….

Signature of the Student: Signature of

the Guide

Date : ………………….. Date:

………………….

Enrolment No.:

Name and Address Name,

Designation and

of the Student: Address of the

Guide:

Special Note:

Acknowledgement

I would like to express my sincere gratitude to all those who has supported me with the successful completion of this project, “*Decentralized Voting System Using Blockchain* “.

First and foremost, I thank Graphic Era Deemed to be University and the Department of Computer Applications for providing me with the opportunity and resources to undertake this Project.

I also wish to extend my gratitude to my guide, Ms. Vandana Rawat, for their immense support.

This project has been a valuable learning experience, and I’m truly thankful to everyone who contributed to its successful completion.

## Candidate Name and Signature

**Table of Contents**

|  |
| --- |
| Candidate’s Declaration i |
| Acknowledgements iii |
| Table of Contents iv  List of Figures vi  **Chapter 1. INTRODUCTION 1**   * 1. **Introduction to blockchain 1**   2. **Decentralized voting using blockchain 2**   3. **Brief Explanation of existing system 3**   4. **Disadvantages of existing system 3**   5. **Brief explanation of proposed system 4**   6. **Advantages of Proposed System 4**   7. **Objective and Scope of Decentralized Voting System Using Blockchain 4**   **Chapter 2. SYSTEM ANALYSIS AND REQUIREMENTS SPECIFICATIONS 7**   * 1. **Requirement Analysis 7**   2. **Data Flow Diagram 8**   3. **Use Case Diagram 9**   **Chapter 3. SYSTEM DESIGNS 10**   * 1. **ER Diagram 10**   2. **System Architecture 10**   3. **Flowchart 11**   4. **Algorithm 13**   **Chapter 4. PROJECT MANAGEMENT 15**   * 1. **Project planning and scheduling 15**      1. **Project Development Approach 15**      2. **Project Plan 16** |

* 1. [Risk Management 18](#_TOC_250023)
     1. [Risk Identification 18](#_TOC_250022)
     2. [Risk Analysis 19](#_TOC_250021)
     3. [Risk Planning 20](#_TOC_250020)
  2. [Estimation 21](#_TOC_250019)
     1. [Cost Analysis 21](#_TOC_250018)

[Chapter 5. INPUT DESIGN 25](#_TOC_250017)

* 1. [Modules 25](#_TOC_250016)
  2. [Coding 26](#_TOC_250015)

[Chapter 6. OUTPUT DESIGN 38](#_TOC_250014)

Chapter 7. SYSTEM TESTING , IMPLEMENTATION & MAINTENANCEA 40

* 1. [Types of Testing 40](#_TOC_250013)
     1. [Unit Testing 40](#_TOC_250012)
     2. [Integration Testing 40](#_TOC_250011)
     3. [Functional Testing 41](#_TOC_250010)
     4. [White Box Testing 41](#_TOC_250009)
     5. [Black Box Testing 42](#_TOC_250008)
  2. [Test Results 42](#_TOC_250007)
     1. [Test Case 1 42](#_TOC_250006)
     2. [Test Case 2 43](#_TOC_250005)
     3. [Test Case 3 43](#_TOC_250004)
     4. [Test Case 4 44](#_TOC_250003)
     5. [Test Case 5 45](#_TOC_250002)

[Chapter 8. SUMMARY AND FUTURE SCOPE 46](#_TOC_250001)

[References 47](#_TOC_250000)

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure 1.** | Level 0 data flow diagram | 8 |
| **Figure 2**. | Level 1 data flow diagram | 8 |
| **Figure 3.** | Level 2 data flow diagram | 9 |
| **Figure 4.** | Use case diagram | 9 |
| **Figure 5.** | ER Diagram | 10 |
| **Figure 6.** | System architecture | 10 |
| **Figure7.** | login page | 40 |
| **Figure 8.** | Admin portal | 40 |
| **Figure 9.** | voting page | 41 |

* 1. **Introduction to Blockchain**

# CHAPTER 1 INTRODUCTION

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. **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.

Decentralized voting using Ethereum blockchain is a secure, transparent and tamper-proof way of conducting online voting. It is a decentralized application built on the Ethereum blockchain network, which allows participants to cast their votes and view the voting results without the need for intermediaries.

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.

## 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.

## 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.

## 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.

## 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.
  1. **Objective and Scope of Decentralized Voting System Using Blockchain**

OBJECTIVE:

1. Enhance Transparency: Ensure a transparent voting process where all actions are recorded on an immutable blockchain ledger.
2. Increase Security: Minimize vulnerabilities to tampering, fraud, or hacking through cryptographic security.
3. Promote Trust: Build trust among voters by providing a verifiable and auditable system.
4. Facilitate Accessibility: Enable remote and convenient voting for citizens, ensuring inclusivity.
5. Improve Efficiency: Reduce costs and time associated with traditional voting methods while enhancing operational efficiency.
6. Eliminate Centralized Control: Distribute voting authority to prevent single points of failure or manipulation.

SCOPE:

1. Global Application: Can be applied for local, national, or even international elections or decision-making processes.
2. Integration with Digital IDs: Can integrate with secure digital identity systems for voter authentication.
3. End-to-End Verifiability: Allows voters to verify their vote without compromising anonymity.
4. Immutable Record: Ensures that once a vote is cast, it cannot be altered, ensuring the integrity of the process.
5. Customizable Features: Can include options like instant vote counting, live updates, and customizable voting rules.
6. Smart Contract Automation: Automates election processes, such as counting and validation, using smart contracts.
7. Reduced Costs: Eliminates paper ballots and manual labor, reducing administrative expenses.
8. Auditability: Facilitates easy audits and recounts using the blockchain ledger.

# CHAPTER 2

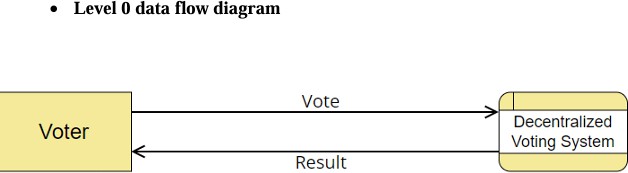
**SYSTEM ANALYSIS AND REQUIREMENTS SPECIFICATIONS**

## 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.

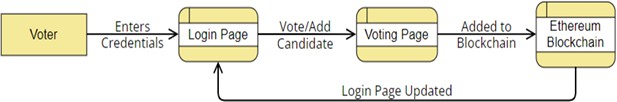
* + - Hardware Requirements
* Processor – 2 GHz or more
* RAM – 4 GB or more
* Disk Space – 100 GB or more
  + - Software Requirements
* Node.js (version – 18.14.0)
* Web3.js (version – 1.8.2)
* Truffle (version – 5.7.6)
* Solidity (version – 0.5.16)
* Ganache (version – 7.7.3)
* Metamask
* Python (version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

## Data Flow Diagram



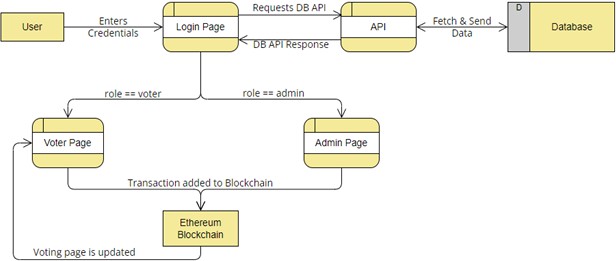
**Figure 1** Level 0 data flow diagram

## Level 1 data flow diagram



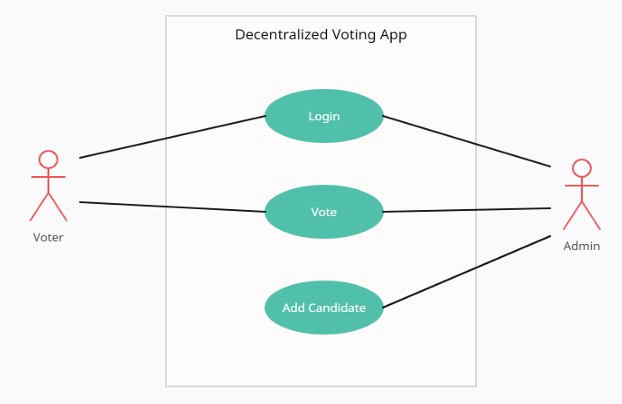
**Figure 2** Level 1 data flow diagram

## Level 2 data flow diagram



**Figure 3** Level 2 data flow diagram

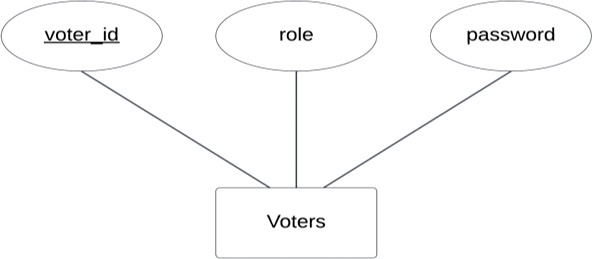
## 2.3 Use Case Diagram



**Figure 4** Use case diagram

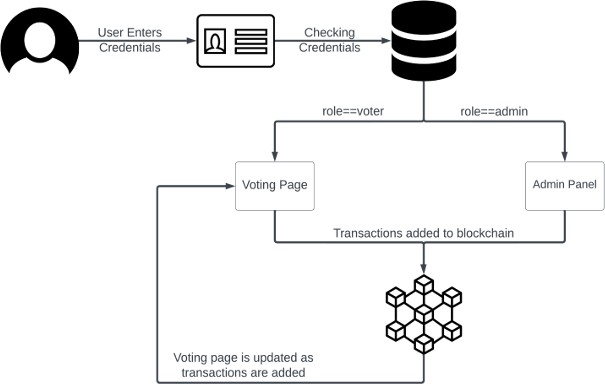
# CHAPTER 3 SYSTEM DESIGNS

## ER Diagram



**Figure 5** ER Diagram

## System Architecture



**Figure 6** 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.

## Flowchart

* + 1. Start
    2. User Authentication

Voter enters ID and password. System verifies credentials.

If valid, proceed.

If invalid, show error and restart.

* + 1. Display Candidates

Fetch the list of candidates from the blockchain.

* + 1. Voting Process

Voter selects a candidate.

Confirm the vote.

* + 1. Record Vote on Blockchain Encrypt the vote.

Store the encrypted vote on the blockchain. Update the total count.

* + 1. End Voting Session
    2. Vote Results

Fetch results from the blockchain.

Display results to the admin or authorized users.

* + 1. End

## Algorithm

1. Initialization:

Connect to the blockchain network. Load smart contract for voting.

1. Authentication:

Input: Voter ID and password. Verify against the database:

If valid, grant access.

If invalid, deny access and request re-entry.

1. Candidate Selection:

Fetch the list of candidates from the blockchain smart contract. Display candidates to the voter.

Record voter's selection.

1. Vote Submission:

Encrypt the selected candidate data.

Add the encrypted vote to the blockchain using the smart contract.

1. Vote Verification:

Smart contract verifies and stores the vote securely. Update the candidate's total votes in the blockchain.

1. End Voting:

Allow the admin to close the voting session by calling a specific function in the smart contract.

1. Result Computation:

Smart contract fetches and aggregates votes for each candidate. Return results to authorized users.

1. End.

# CHAPTER 4 PROJECT MANAGEMENT

## Project planning and scheduling

* + 1. **Project Development Approach**

1. Agile Development

The project follows an agile development methodology, enabling iterative progress through regular feedback and flexibility to changes.

Sprints: The project is divided into small sprints (e.g., 2 weeks). Team Roles:

Developer: Responsible for coding smart contracts and UI. Tester: Ensures system functionality and security.

Project Manager: Oversees the timeline and deliverables.

1. Tools and Technologies

Blockchain: Ethereum for implementing the decentralized ledger. Programming Languages: Solidity (smart contracts), JavaScript (frontend). Frameworks: Truffle for testing and deploying smart contracts.

Database: IPFS for decentralized data storage.

1. Phased Development

Phase 1: Requirements gathering and system design. Phase 2: Smart contract development and integration. Phase 3: Frontend and backend development.

Phase 4: Testing and deployment.

## Project Plan

1 .Timeline

Week 1–2: Requirement Analysis and Feasibility Study

Week 3–4: Designing the Blockchain Architecture and Smart Contracts Week 5–6: Development of Smart Contracts and Testing

Week 7–8: Integration with Frontend and Backend Week 9–10: Testing and Bug Fixes

Week 11: Deployment on Ethereum Mainnet/Testnet

1. Key Milestones

Completion of system architecture.

Deployment of the first version of smart contracts. Functional testing of the voting process.

Final deployment and presentation.

1. Resource Allocation

Developers: Write and test smart contracts. Testers: Validate secure voting.

Project Lead: Monitor progress.

1. Deliverables

Secure and transparent voting system. Comprehensive project report and documentation.

## Risk Management

## Risk Identification

1. Technical Risks

Bugs in smart contract code.

Blockchain network downtime or congestion.

Security vulnerabilities leading to potential hacking or tampering.

1. Operational Risks

Delays in project deliverables due to team miscommunication. Resource unavailability, such as skilled developers or testing tools. Insufficient testing or oversight.

1. External Risks

Regulatory changes affecting the use of blockchain for voting.

Lack of adoption by users due to unfamiliarity with blockchain technology. Dependency on third-party platforms like Ethereum and IPFS.

## Risk Analysis

1. Likelihood and Impact Matrix High Likelihood, High Impact:

Bugs in smart contracts (critical risk). Security vulnerabilities (critical risk).

Medium Likelihood, Medium Impact:

Network congestion during peak voting hours. Team delays or resource unavailability.

Low Likelihood, High Impact:

Regulatory changes.

1. Risk Priority

Priority 1: Addressing critical security vulnerabilities. Priority 2: Ensuring thorough testing to reduce bugs.

Priority 3: Mitigating network congestion and operational delays.

## Risk Planning

1. Mitigation Strategies Technical Risks:

Conduct rigorous testing of smart contracts using tools like MythX. Implement multi-layer encryption to secure voter data.

Set up fallback systems to handle network congestion.

Operational Risks:

Use agile methodology for regular progress tracking and team communication. Allocate backup resources to ensure availability.

External Risks:

Stay updated on regulations and adapt the system as necessary. Conduct training sessions to educate users on blockchain voting.

1. Contingency Plans

Deploy an alternate blockchain (e.g., Binance Smart Chain) in case of Ethereum failures.

Extend project timelines to accommodate unexpected delays or issues. Implement manual voting mechanisms as a fallback for critical failures.

1. Monitoring and Review

Assign a dedicated risk manager to track risks throughout the project. Schedule weekly reviews to reassess risk likelihood and impact.

Maintain a risk register to document risks and resolutions.

## Estimation

## Cost Analysis

* + - * Development Costs

Smart Contract Development:

Developer salaries (based on hours or project rate).

Tools and platforms used (e.g., Truffle, Remix, Solidity development environment).

Testing tools and frameworks (e.g., MythX, Ganache).

Frontend and Backend Development:

Developer time for frontend UI/UX development (React, HTML, CSS, JavaScript).

Backend integration for blockchain connection (e.g., Web3.js).

Design tools (e.g., Adobe XD, Figma) for user interface design.

* + - * Infrastructure and Hosting Costs

Blockchain Network:

Transaction fees on the Ethereum network (gas fees for deploying smart contracts and votes).

Deployment on Ethereum Testnet or Mainnet (could vary based on transaction volume).

Web Hosting:

Server costs for hosting the voting platform (e.g., AWS, DigitalOcean, or Firebase).

Storage costs for decentralized storage on IPFS.

Maintenance and Scaling:

Ongoing server costs.

Monitoring and scaling costs for handling user traffic.

* + - * Operational Costs

Team Salaries:

Project manager, developers, testers, UI/UX designers.

Payments could be monthly or based on the project timeline.

Consulting/External Costs:

Hiring external security experts for audit of smart contracts and system security.

Training and Education:

Training costs for users and stakeholders to ensure familiarity with the blockchain- based voting system.

* + - * Contingency Costs

Buffer for Unexpected Issues:

A portion of the budget should be reserved for unforeseen problems like system errors, regulatory issues, or other unplanned expenses.

* + - * Total Esti 4mated Budget

Fixed Costs: Costs that remain constant throughout the project, such as software licenses and basic infrastructure.

Variable Costs: Costs that change based on resource utilization, like developer hours, transaction fees, and server load.

# CHAPTER 5 INPUT DESIGN

## 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:

* 1. Voters can securely authenticate themselves to access the voting system using their unique credentials.
  2. Voters can access information about the candidates running for various positions, such as their names, parties, and other relevant details.
  3. Voters can verify the status of their votes and ensure that their choices are accurately recorded in the blockchain.

1. 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:

* 1. 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.
  2. Admin can manually verify the candidate and can start the voting process.

## Coding

1. Migrations. Sol pragma solidity ^0.5.15;

contract Migrations { address public owner;

unit public last\_completed\_migration;

modifier restricted() {

require(msg.sender == owner, "Access restricted to owner");

\_;

}

constructor() public { owner = msg.sender;

}

function setCompleted(unit completed) public restricted { last\_completed\_migration = completed;

}

function upgrade(address new\_address) public restricted { Migrations upgraded = Migrations(new\_address); upgraded.setCompleted(last\_completed\_migration);

}

}

1. 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,ca ndidates[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);

}

1. App.js

4. //import "../css/style.css"

5.

6. const Web3 = require('web3');

7. const contract = require('@truffle/contract'); 8.

9. const votingArtifacts = require('../../build/contracts/Voting.json');

10.var VotingContract = contract(votingArtifacts) 11.

12.window.App = {

1. eventStart: function() {
2. window.ethereum.request({ method: 'eth\_requestAccounts' });
3. VotingContract.setProvider(window.ethereum)
4. VotingContract.defaults({from: window.ethereum.selectedAddress,gas:6654755})
5. // Load account data
6. App.account = window.ethereum.selectedAddress;
7. $("#accountAddress").html("Your Account: " + window.ethereum.selectedAddress);
8. VotingContract.deployed().then(function(instance){
9. instance.getCountCandidates().then(function(countCandidates)

{

1. $(document).ready(function(){
2. $('#addCandidate').click(function() {
3. var nameCandidate = $('#name').val();
4. var partyCandidate = $('#party').val();
5. instance.addCandidate(nameCandidate,partyCandida te).then(function(result){ })

30. });

1. $('#addDate').click(function(){
2. var startDate =

Date.parse(document.getElementById("startDate").value)/1000;

1. var endDate

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

1. instance.setDates(startDate,endDate).then(funct ion(rslt){

37.

38.

39.

40.

41.

42.

console.log("tarihler verildi");

});

});

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

1. var startDate = new Date(result[0]\*1000);
2. var endDate = new Date(result[1]\*1000); 45.
3. $("#dates").text( startDate.toDateString(("#DD#/#MM#/#YYYY#")) + " - " + endDate.toDateString("#DD#/#MM#/#YYYY#"));
4. }).catch(function(err){
5. console.error("ERROR! " + err.message)

49. });

50. });

51.

1. for (var i = 0; i < countCandidates; i++ ){
2. instance.getCandidate(i+1).then(function(data){
3. var id = data[0];
4. var name = data[1];
5. var party = data[2];
6. var voteCount = data[3];
7. 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>"

1. $("#boxCandidate").append(viewCandidates)

60. })

61. }

62.

63. window.countCandidates = countCandidates

64. });

65.

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

71. }

72. });

73.

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

76. })

77. },

78.

1. vote: function() {
2. var candidateID = $("input[name='candidate']:checked").val();
3. if (!candidateID) {
4. $("#msg").html("<p>Please vote for a candidate.</p>")
5. return

84. }

85. VotingContract.deployed().then(function(instance){

|  |  |  |
| --- | --- | --- |
| 86. |  | instance.vote(parseInt(candidateID)).then(function(result){ |
| 87. |  | $("#voteButton").attr("disabled", true); |
| 88. |  | $("#msg").html("<p>Voted</p>"); |
| 89. |  | window.location.reload(1); |
| 90. |  | }) |
| 91. |  | }).catch(function(err){ |
| 92. |  | console.error("ERROR! " + err.message) |
| 93. |  | }) |
| 94.  95.}  96. | } |  |
| 1. window.addEventListener("load", function() { 2. if (typeof web3 !== "undefined") { 3. console.warn("Using web3 detected from external source like Metamask") 4. window.eth = new Web3(window.ethereum) 5. } else { 6. 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"](http://truffleframework.com/tutorials/truffle-and-metamask)) 7. window.eth = new Web3(new Web3.providers.HttpProvider("http://127.0.0.1:9545")) | | |
| 104. | } | |
| 105. | window.App.eventStart() | |
| 106. | }) | |
| 107. |  | |

1. 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=${voter\_id}&password=${passwor d}`, { 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=B

earer ${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=Be arer ${localStorage.getItem('jwtTokenVoter')}`);

}

})

.catch(error => {

console.error('Login failed:', error.message);

});

});

1. 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"

)

5. Package.json

6. {

1. "name": "decentralized-voting",
2. "version": "1.0.0",
3. "description": "",
4. "main": "index.js",
5. "scripts": {
6. "start": "truffle develop"

13. },

1. "author": "",
2. "license": "ISC",
3. "dependencies": {
4. "jsonwebtoken": "^9.0.0",
5. "@truffle/contract": "^4.6.18",
6. "browserify": "^17.0.0",
7. "dotenv": "^16.0.3",
8. "express": "^4.18.2",
9. "web3": "^1.9.0"

23. }

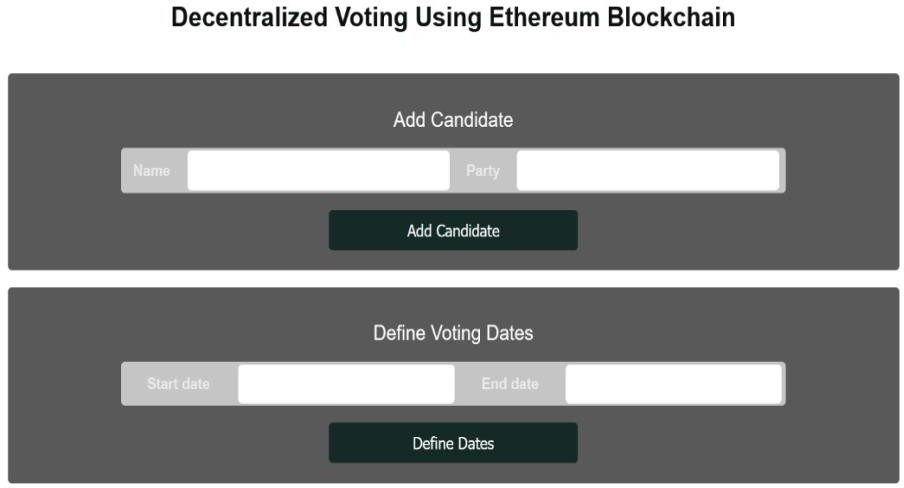
24.}

25.

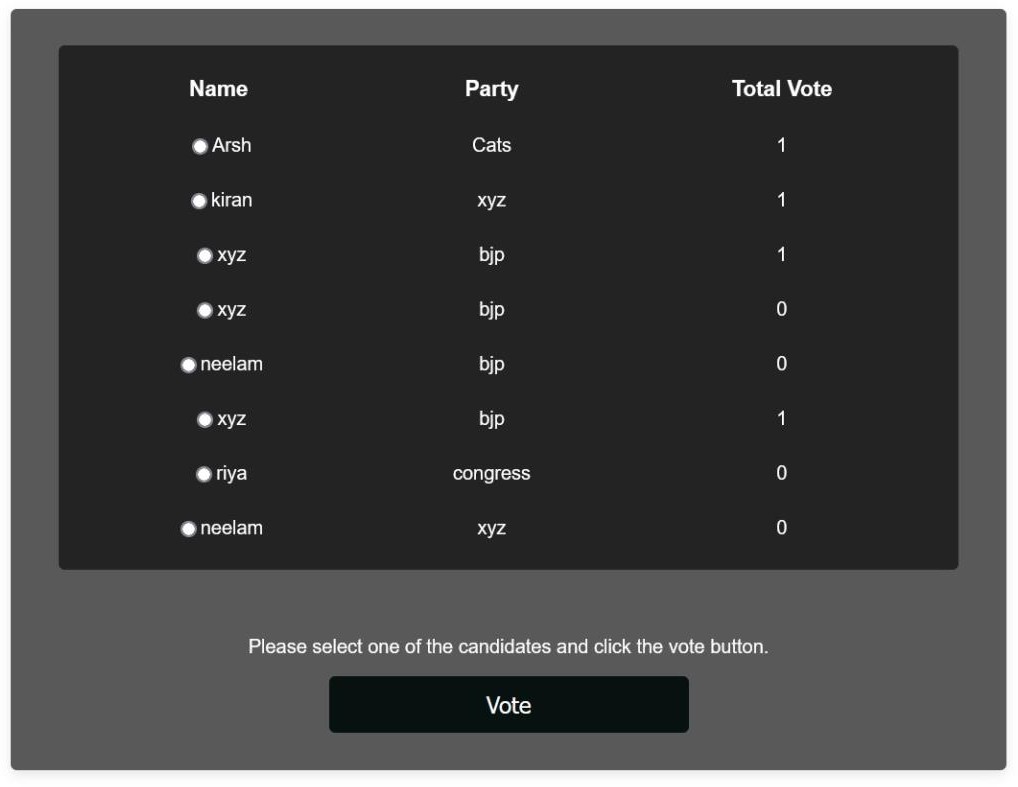
# CHAPTER 6 OUTPUT DESIGN



**Figure7** login page



**Figure 8** Admin portal



**Figure 9** voting page

# CHAPTER 7

**SYSTEM TESTING , IMPLEMENTATION & MAINTENANCEA**

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

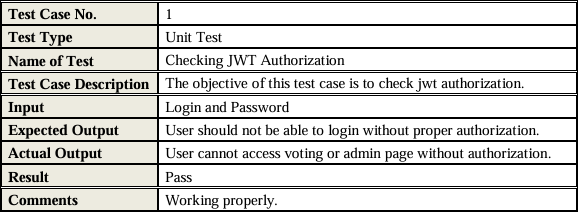
White box testing, also known as structural testing or glass-box testing, is a type of testing that examines the internal structure and implementation of a software system. It involves testing the code itself and checking that it is functioning correctly and adhering to coding standards. This type of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

## Black Box Testing

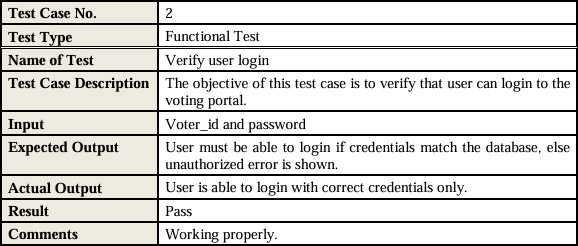
Black box testing, also known as functional testing, is a type of testing that examines the external behavior and interfaces of a software system. It involves testing the system from the user's perspective, without looking at the internal structure or implementation, and checking that it is functioning correctly and meeting the requirements. This type of testing helps to identify and resolve issues related to usability, compatibility, and performance.

## Test Results

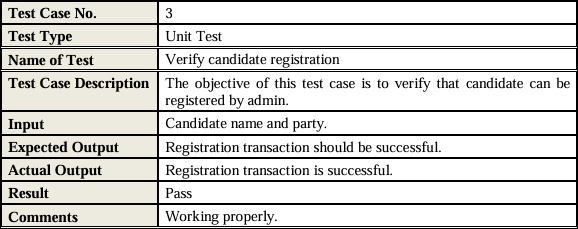
## Test Case 1



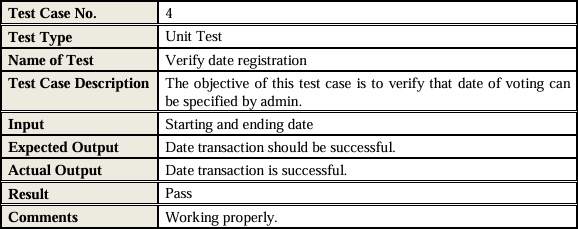
## Test Case 2



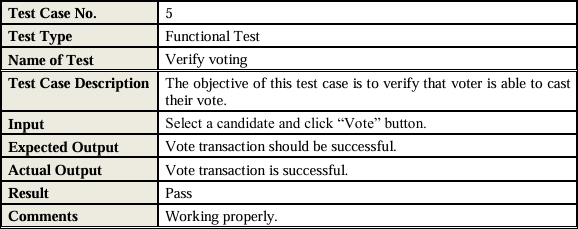
## Test Case 3



## Test Case 4



## Test Case 5



# CHAPTER 8 SUMMARY 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 Enhancement:

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.

# REFERENCES

1. Ethereum Documentation

Ethereum Foundation. (2024). Ethereum: A Decentralized Platform. Retrieved from https://ethereum.org/en/developers/docs/

1. Blockchain Technology

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf

1. Smart Contract Development

Solidity Documentation. (2024). Solidity: Smart Contract Programming Language. Retrieved from https://soliditylang.org/docs/

1. Decentralized Voting Systems

Arya, A., & Kumar, A. (2023). Decentralized Voting Systems Using Blockchain Technology. Journal of Blockchain Research, 10(2), 56-64.

1. Security Audits for Blockchain

Kuleshov, M., & Moloch, C. (2022). Security Audits for Blockchain-based Smart Contracts. Security in Blockchain, 8(1), 100-112.

1. Risk Management in IT Projects

Hillson, D., & Simon, P. (2019). Practical Project Risk Management: The ATOM Methodology. CRC Press.

1. Cost Estimation in Software Projects

Boehm, B. W. (1981). Software Engineering Economics. Prentice Hall.

1. IPFS for Decentralized Storage

Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System. Retrieved from https://ipfs.io/