A Mini Project Report (CS755PC)

on

Decentralized Voting System Using Blockchain

Submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

In

COMPUTER SCIENCE & ENGINEERING

(Artificial Intelligence & Machine Learning)

by

Ms. Sanala Vaishnavi

(22265A6602)

Under the guidance of

Ms. D. Deepika

Assistant Professor

DEPARTMENT OF EMERGING TECHNOLOGIES

Mahatma Gandhi Institute of Technology

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)

Kokapet(V), Gandipet(M), Hyderabad.

Telangana - 500 075.

2024 – 2025

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)

GANDIPET, HYDERABAD – 500075, Telangana

CERTIFICATE

This is to certify that the project entitled **"Decentralized Voting using Blockchain"** is being submitted by **SANALA VAISHNAVI** bearing Roll No. **22265A6602** in partial fulfilment of the requirements for the Mini Project (CS755PC) in **COMPUTER SCIENCE AND ENGINEERING(Artificial Intelligence & Machine Learning)** is a record of bonafide work carried out by her.

The results of the investigations enclosed in this report have been verified and found satisfactory.

Project Guid	Head of the Department	Principal
Ms. D. Deepika	**Dr. M. Rama Bai**	**Prof. G Chandra Mohan Reddy**
Assistant Professor	Professor and Hod	Professor

External Examiner

DECLARATION

This is to certify that the work reported in this project titled **—DECENTRALIZED VOTING SYSTEM USING BLOCKCHAIN"** is a record of work done by me in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by me and not copied from any other source.

SANALA VAISHNAVI (22265A6602)

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to **Prof. G Chandra Mohan Reddy , Principal MGIT**, for providing the working facilities in college.

I wish to express my sincere thanks and gratitude to **Dr. M. Rama Bai, Professor and HoD**, Department of ET, MGIT, for all the timely support and valuable suggestions during the period of project.

I am extremely thankful to **Ms. Mahammad Nazni, Professor** and **Mr. Mallela Srikanth, Associate Professor,** Department of ET, MGIT, mini project coordinators for their encouragement and support throughout the project.

I am extremely thankful and indebted to my internal guide **Ms. D. Deepika**, **Assistant Professor**, Department of ET, for her constant guidance, encouragement and moral support throughout the project.

Finally, I would also like to thank all the faculty and staff of CSE Department who helped me directly or indirectly, for completing this project.

SANALA VAISHNAVI (22265A6602)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Decentralized voting using the Ethereum blockchain offers a secure, transparent, and tamper-proof method for conducting online elections. Unlike traditional systems, it eliminates the need for intermediaries by leveraging blockchain technology to provide a reliable and decentralized platform for voting. This approach ensures that votes are securely recorded on the blockchain, making it virtually impossible to manipulate or alter results.

At the heart of this system lies smart contracts, which automate and enforce the voting process. These contracts are programmed to handle critical aspects, such as vote casting, validation, and result computation, without human intervention. This not only reduces the risk of errors but also ensures transparency and fairness, as all transactions are publicly verifiable on the blockchain. Participants can securely cast their votes and view the results, knowing that their data is protected and the process is unbiased.

By utilizing blockchain technology, decentralized voting provides a cost-effective and trustworthy solution for elections. It addresses common challenges like fraud, tampering, and lack of transparency associated with traditional voting systems. Whether for large-scale political elections or smaller organizational votes, this innovative approach can revolutionize the way we conduct elections, fostering greater confidence in democratic processes.

1. INTRODUCTION

A decentralized voting system utilizing blockchain technology introduces a groundbreaking way of conducting elections securely, transparently, and efficiently. Unlike traditional voting methods, which rely on centralized authorities to manage and validate the process, blockchain-based systems distribute these tasks across a decentralized network. This eliminates the need for intermediaries, ensuring that each vote is securely recorded and counted without the risk of tampering or manipulation. By leveraging the principles of decentralization, this system provides a robust solution to address common challenges in electoral processes, such as fraud, lack of transparency, and inefficiencies.

At the core of this system lies blockchain technology, which acts as a digital ledger to record votes. Each vote is treated as a transaction, encrypted, and added to the blockchain as an immutable record. Blockchain's decentralized structure ensures that this data is replicated across multiple nodes in the network, making it resistant to hacking or unauthorized alterations. Additionally, smart contracts, which are self-executing programs stored on the blockchain, automate crucial aspects of the voting process, such as vote validation, adherence to rules, and result computation. This automation minimizes human intervention, ensuring a transparent and error-free election process.

One of the significant advantages of decentralized voting systems is their ability to provide transparency while maintaining voter privacy. Every participant can verify the process and results on the blockchain, fostering trust and accountability. At the same time, advanced cryptographic techniques ensure voter anonymity, preventing unauthorized access to sensitive information. Moreover, the decentralized architecture makes the system resilient to failures, as there is no single point of failure. Even if some nodes in the network are compromised, the system remains operational and secure.

As a modern solution, blockchain-based voting systems have the potential to revolutionize democratic processes. By addressing issues such as voter fraud, tampering, and inefficiencies, this approach offers a reliable alternative to traditional methods. Furthermore, the scalability and adaptability of blockchain technology make it suitable for various types of elections, from small organizational votes to large-scale national elections. In the digital era, decentralized voting represents a significant step forward in creating fair, secure, and accessible electoral systems.

1.1. Motivation

The motivation for developing a decentralized voting system using blockchain stems from the need to overcome the limitations of traditional voting methods, such as fraud, tampering, inefficiency, and lack of transparency. Conventional systems rely on centralized authorities, making them vulnerable to manipulation and cyberattacks. Blockchain technology addresses these challenges by providing a secure, decentralized, and immutable platform where votes are transparently recorded and cannot be altered without consensus. Its distributed ledger ensures accountability, allowing participants to verify the voting process and results in real-time, reducing doubts about fairness. Smart contracts further enhance reliability by automating vote validation and result computation, minimizing errors and biases. Additionally, blockchain enables secure online voting, making elections more accessible to remote and international voters while maintaining data security and voter anonymity. This combination of security, transparency, and inclusivity offers a scalable and transformative solution to modernize electoral processes, fostering public trust and strengthening democratic systems.

1.2 Problem Definition

Traditional voting systems face significant challenges such as vote tampering, voter fraud, lack of transparency, and reliance on centralized authorities, all of which undermine the integrity of election outcomes. These systems are vulnerable to cyberattacks, data breaches, and manipulation, compromising the reliability of the electoral process. Additionally, the manual handling of votes can lead to delays, errors, and increased costs, reducing efficiency and causing disputes. Accessibility remains another issue, as remote voters often struggle to participate due to logistical barriers, reducing voter turnout and election legitimacy. Furthermore, existing online voting systems lack the transparency and security needed to build public trust. To address these problems, a blockchain-based decentralized voting system offers a solution by ensuring tamper-proof vote recording, real-time verifiability, and enhanced accessibility for all voters. By leveraging blockchain's decentralization, immutability, and transparency, this approach aims to create a secure, transparent, and efficient electoral process, strengthening trust and participation in democratic systems.

1.3 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 . The main drawbacks of existing systems include:

- **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.

- **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.

- **Slow results:** Counting paper ballots can be a time-consuming and labor-intensive process, which can delay the announcement of election results.

- **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.

- **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.

- **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.

1.4 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.

1.4.1 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.

1.4.2 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.

1.5 Requirements Specification

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.

1.5.1 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)

1.5.2 Hardware Requirements

- Processor – 2 GHz or more
- RAM – 4 GB or more

- Disk Space – 100 GB or more

1.5.1.1 Node.js (Version – 18.14.0)

Node.js is essential for handling the JavaScript logic and interaction with the frontend and blockchain. Once installed, it acts as the backend runtime environment, where you'll manage dependencies, such as Web3.js and Truffle, and implement the logic to interact with smart contracts. You can also use Node.js to build an API layer or middleware to facilitate communication between the frontend (HTML/JS) and the blockchain.

1.5.1. 2 Web3.js (Version – 1.8.2)

Web3.js provides the necessary tools to interact with Ethereum smart contracts. After installing Web3.js, you can use it in your JavaScript files to initialize connections to Ethereum networks (e.g., Ganache for local testing or the mainnet for deployment). Using Web3.js, you can deploy contracts, read blockchain data (e.g., voter registration), and send transactions such as vote casting. It is also responsible for integrating MetaMask to sign transactions securely.

1.5.1.3 Truffle (Version – 5.7.6)

Truffle is used to streamline the development of Ethereum-based smart contracts. Once installed, you can initialize a new Truffle project, where you will write your smart contract logic in Solidity. Truffle provides commands to compile contracts, migrate (deploy) them to blockchain networks, and test them. Truffle's built-in testing framework is useful for simulating interactions with smart contracts to ensure that everything functions as expected before deployment.

1.5.1.4 Solidity (Version – 0.5.16)

Solidity is the language used to write the smart contract for the decentralized voting system. The contract will include the logic for registering voters, casting votes, and tallying results. Solidity 0.5.16 is the specified version, and the syntax and features are aligned with that version. You must ensure that the pragma directive in your Solidity code corresponds to this version (pragma solidity ^0.5.16;) to avoid compatibility issues. Once written, the contract is compiled and deployed via Truffle or other compatible tools.

1.5.1.5 Ganache (Version – 7.7.3)

Ganache provides a local Ethereum blockchain to deploy your contracts during development. After installation, Ganache allows you to create an isolated Ethereum network that simulates blockchain transactions. It provides a user-friendly interface to monitor transactions, check balances, and track smart contract interactions in real-time. You can configure Ganache's RPC settings (usually http://127.0.0.1:7545) in the Truffle truffle-config.js to connect Truffle to your local Ganache instance for contract deployment.

1.5.1.6 MetaMask

MetaMask is the bridge between your decentralized application (dApp) and the user's browser wallet. It allows users to securely manage their Ethereum-based accounts and sign transactions without exposing private keys. In the decentralized voting system, MetaMask will allow users to connect their Ethereum account to the application, authenticate their identity, and authorize the sending of transactions (such as casting votes). You will need to configure the connection to the local Ganache instance or an Ethereum testnet/mainnet via MetaMask.

1.5.1.7 Python (Version – 3.9)

Python serves as the backend language for handling business logic, managing data, and interacting with the MySQL database. Python 3.9 is the specified version, and you can use it to develop APIs that interact with your database and potentially the smart contract logic (for example, verifying voters before allowing them to vote). It is also used for running backend services like user authentication and managing the voting data lifecycle.

1.5.1.8 FastAPI

FastAPI is used for creating the RESTful API that powers your decentralized voting system's backend. It is fast, asynchronous, and built with Python. After installing it via pip, you can build API endpoints to handle tasks like fetching voter data, registering new voters, or showing the voting status. FastAPI is designed to work seamlessly with Python 3.9 and is highly efficient, which helps

in handling real-time interactions between the user interface and the backend server. You can use uvicorn to run the FastAPI server, providing an interface for your frontend to interact with.

1.5.1.9 MySQL Database (Port – 3306)

MySQL is the relational database used to store non-blockchain data. While blockchain stores the immutable voting data, MySQL will handle other application data, such as user profiles, session states, and vote histories. You will need to design a schema to handle this data, and Python's pymysql or mysql-connector can be used to integrate the database with the backend FastAPI. The default MySQL port is 3306, and you must configure your database connection strings properly.

2. LITERATURE SURVEY

1. 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.

2. 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.

3. 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.

4. 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. Evoting 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.

5. 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.

Table 1: Comparison of Literature survey

Section	Paper	Focus Area	Challenges Addressed	Proposed Solutions/Key Points	Future Directions/Findings
1.	**Title:** Online Voting System Using Blockchain **Author:** Vaibhav Anasune, Pradeep Choudhari, Madhura Kelapure, Pranali Shirke, Prasad Halgaonkar	Survey of traditional voting systems and their security issues.	Manipulation of centralized database, lack of transparency in traditional systems.	Highlights security risks in centralized systems and explores blockchain-based voting benefits.	Suggests the necessity of advanced security methods and decentralized systems to improve trust and transparency in voting.
2.	**Title:** A Systematic Literature Review and Meta-Analysis on Scalable Blockchain-Based Electronic Voting Systems **Author:** Uzma Jafar et al.	Scalability in blockchain-based voting systems.	Issues of scalability, authentication, data integrity, and cost.	Conducted an SLR of 76 papers; explores cryptographic solutions, performance parameters, and verification methods.	Proposes future research avenues in scalable voting systems, emphasizing cost-effective solutions while maintaining performance, transparency, and security.
3.	**Title:** A Survey of Blockchain Based on E-voting Systems **Author:** Yousif Osman Abuidris, Rajesh Kumar, Wang Wenyong	Security and privacy challenges in blockchain-based e-voting.	Privacy concerns, trust in data storage, and consensus	Highlights blockchain's tamper-proof nature and compares existing systems for security/privacy requirements.	Identifies security/privacy gaps and the importance of adopting tailored blockchain solutions for e-voting systems.

			mechanisms.		
4.	**Title:** Survey on Voting System Using Blockchain Technology **Author:** Mayur Shirsath et al.	Blockchain as a solution for secure online voting.	Physical voting manipulation, remote accessibility, and lack of technical simplicity.	Describes a private blockchain system that encrypts votes as transactions and ensures transparency and accessibility.	Highlights potential for increased voter turnout and secure voting, with recommendations for user-friendly designs to encourage adoption in real-world scenarios.
5.	**Title:** A Survey on Smart Electronic Voting System Using Blockchain Technology **Author:** Naina Nagesh Dhepe and Dr. Pathan Mohd Shafi	Electronic voting in India with blockchain integration.	Security holes in electronic voting systems, challenges in user authentication, and machine protection.	Proposes a high-security blockchain-compatible voting machine to enhance election activities and reduce costs.	Recommends further research into secure authentication methods and integration of blockchain to ensure tamper-proof voting systems, particularly for large-scale democracies like India.

3.METHODOLOGY

The decentralized nature of blockchain ensures fairness and prevents tampering while cryptographic techniques maintain voter anonymity. The system is designed using a combination of smart contracts, secure data handling, and robust cryptographic algorithms for optimal performance in a voting context. Steps followed for the project is as given below:

1. Candidate Registration

2. Preprocessing and Feature Extraction

3. Storing in Database

4. Transaction Process

1. Candidate Registration

Candidate information such as name, address, and other specific details, along with a facial capture for registration in the blockchain-based voting system, is securely stored in a database. This process ensures the integrity and transparency of the electoral process, with candidate details including identification, eligibility, and platform stored as encrypted records within the decentralized ledger. This setup allows for efficient and auditable access to information while maintaining the immutability and security features inherent to blockchain technology.

2. Preprocessing and Feature Extraction

Initially, facial detection algorithms like Haar cascades or deep learning models are employed to locate and identify faces within captured images. Subsequently, normalization techniques ensure that all facial images are resized and oriented consistently for uniformity. To enhance data quality, noise reduction filters are applied, refining the images for clarity. Further, facial alignment algorithms adjust key landmarks (e.g., eyes, nose, mouth) to a standardized position. For feature extraction, geometric measures such as distances between landmarks, texture features like local binary patterns, and deep learning-based features extracted from pre-trained CNN models are utilized.

3. Storing in Database

Storing facial features and candidate information securely in a database for future login purposes within a blockchainbased voting system is crucial for maintaining the integrity and accessibility of user data. By storing encrypted facial features and candidate details (such as name, address, and eligibility) in the database, the system can authenticate users during subsequent logins, ensuring secure access to voting functionalities. Additionally, enabling the admin to create election polls and define political parties within the system provides flexibility and customization for electoral processes. These election-related configurations, along with associated data like candidate lists and voting options, are stored in the database to facilitate efficient management and execution of elections. This approach enhances the overall functionality and reliability of the blockchain-based voting platform while safeguarding sensitive user and election-related information.

4. Transaction Process

First, when a voter logs in using their credentials (which could include facial recognition for candidate verification), the system verifies their identity against stored data in the database. Upon successful authentication, the voter can access the ballot and select their preferred candidate or vote on specific issues. Once a vote is cast, the transaction details, including the voter's identity and chosen options, are bundled into a block. This block is then broadcasted to the network of nodes (computers participating in the blockchain network) for verification. The nodes validate the transaction based on predefined consensus rules. If the transaction is valid, the new block containing the vote is added to the blockchain, ensuring immutability and transparency of the voting record.

3.1 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.

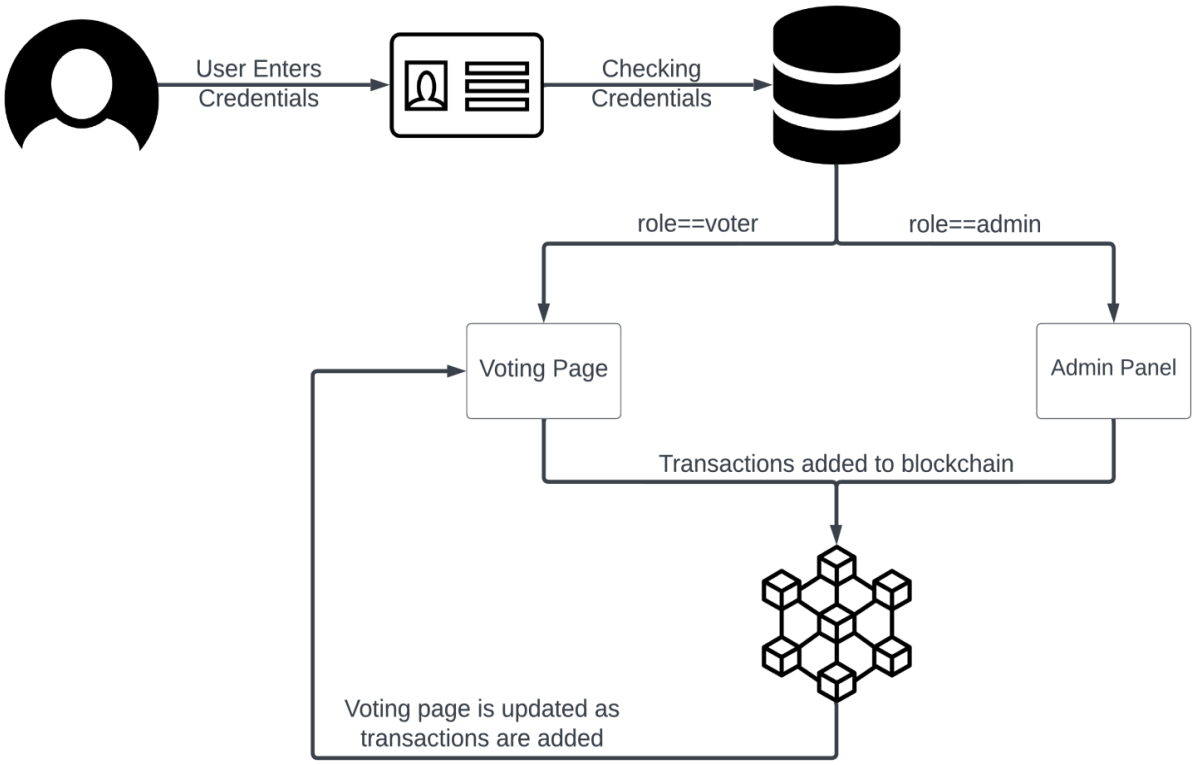

Figure 1 System architecture

In Figure 1 system architecture provides a comprehensive overview of a blockchain-based decentralized voting platform designed for transparency, security, and efficiency. The process begins with users entering their credentials, which are then validated during the Checking Credentials phase. This step ensures that the identity and role of the user are verified, categorizing them either as a voter or an administrator. Depending on the role, users are directed to different parts of the system. Voters are taken to the Voting Page, where they can securely cast their votes, while administrators are directed to the Admin Panel, which provides tools for monitoring and overseeing the voting process.

The Voting Page serves as the interface where voters make their selections. Once a vote is submitted, it is encrypted and recorded as a transaction on the blockchain, ensuring immutability and transparency. Real-time updates on the Voting Page allow voters to confirm that their participation has been successfully recorded, fostering trust in the system. On the other hand, the Admin Panel allows administrators to oversee the election process. While they can monitor the

transactions and ensure the system's integrity, the blockchain's immutable nature prevents any tampering with the recorded votes, maintaining the election's fairness and credibility.

At the core of the architecture is blockchain integration, which ensures the secure storage of votes in a decentralized ledger. Each transaction is cryptographically secured, preserving voter anonymity while guaranteeing data integrity. This setup leverages blockchain technology to provide a tamper-proof and transparent system where every action is permanently recorded. The voting platform incorporates role-based access control, ensuring that voters and administrators can only access functionalities relevant to their roles. Additionally, the system dynamically updates the Voting Page in real-time as transactions are processed, promoting transparency and keeping all stakeholders informed of the latest developments.

This architecture highlights the system's robustness, combining blockchain technology with secure user authentication and role-based access. It offers a secure, auditable, and efficient solution for conducting elections, where transparency and fairness are paramount. By integrating real-time updates, cryptographic protection, and decentralized data handling, the architecture ensures that the voting process is trustworthy and resistant to tampering or manipulation.

3.2 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.

3.3 Data Flow Diagram

Level 0 data flow diagram

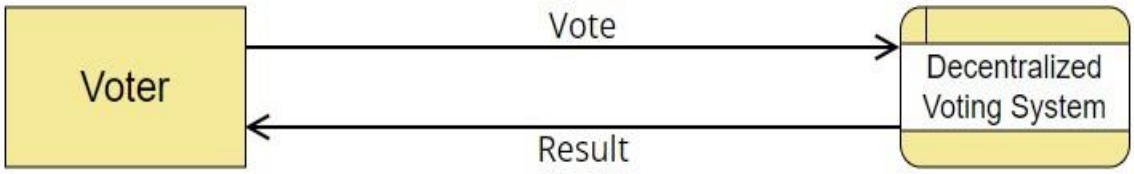

Figure 2 Level 0 Data Flow Diagram

In figure 2 Level 0 Data Flow Diagram represents the interaction between a voter and a decentralized voting system, highlighting the secure and streamlined process of casting and recording votes using blockchain technology. The process begins with the voter submitting their vote through the system's interface. This vote is encrypted to ensure security and then transmitted to the decentralized voting system. The system, built on blockchain, verifies the validity of the vote and ensures it is uniquely registered to prevent issues such as double voting or fraud. Once verified, the vote is recorded on the blockchain, which guarantees immutability and transparency. The system employs cryptographic techniques to maintain the anonymity of the voter while ensuring the integrity of the process. After the vote is successfully recorded, the system sends an acknowledgment back to the voter. Depending on the design, the voter may also view real-time aggregated voting results securely retrieved from the blockchain. This architecture demonstrates how decentralized systems enhance the transparency, security, and trustworthiness of the voting process while safeguarding voter privacy.

Level 1 data flow diagram

The figure 3 illustrates a blockchain-based voting system leveraging Ethereum for security and transparency. The process begins with the voter, who enters their credentials on the Login Page for authentication. Once logged in, the voter is redirected to the Voting Page, where they can either cast their vote for a candidate or add a new candidate if the system permits. After submitting their input, the transaction (vote or candidate addition) is recorded and securely added to the Ethereum blockchain, ensuring that the data is immutable, transparent, and tamper-proof due to the decentralized nature of blockchain technology. Once the voting process is completed, the Login Page is updated to reflect the voter's status, preventing duplicate votes and ensuring the integrity of the election. This system showcases a secure and decentralized approach to managing voting processes, enhancing trust and reliability in election systems.

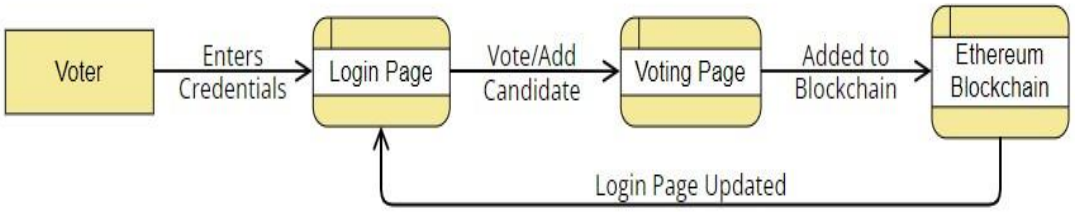

Figure 3 Level 1 Data Flow Diagram

Level 2 data flow diagram

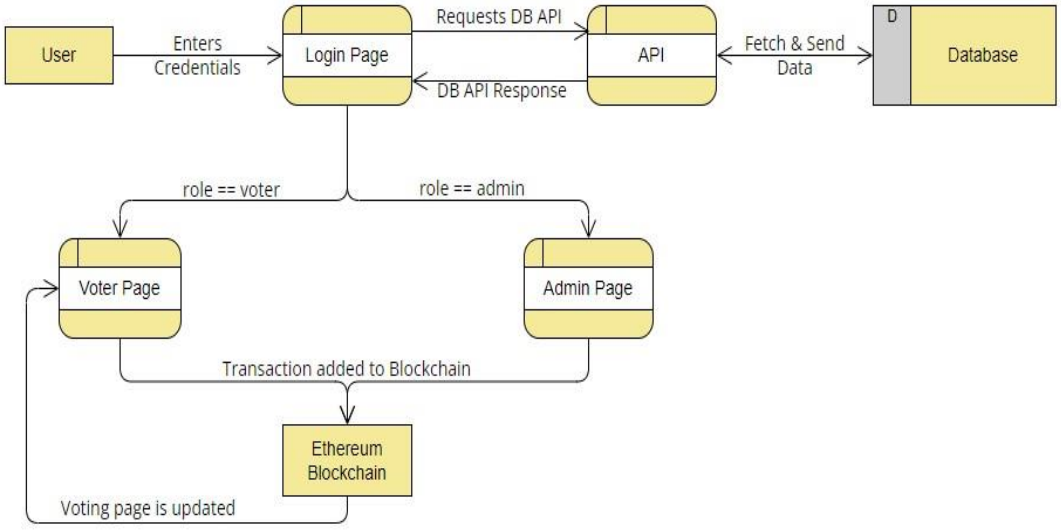

Figure 4 Level 2 Data Flow Diagram

The figure 4 represents a blockchain-based system with user roles for voting and administration. The process begins with the User entering their credentials on the Login Page, which sends a request to the Database API to verify and fetch the user's information. The API interacts with the Database to retrieve or update data as needed and sends a response back to the Login Page. Based on the user's role, they are redirected to either the Voter Page or the Admin Page.

If the user is a voter, they proceed to the Voter Page to cast a vote, and the transaction is securely added to the Ethereum Blockchain, ensuring transparency and immutability. If the user is an administrator, they access the Admin Page, which might include functionalities like managing candidates or overseeing the voting process. Regardless of the user role, any transactions or updates made are reflected back in the system, ensuring the Voting Page is updated dynamically. By integrating Ethereum blockchain technology, this system provides a decentralized and secure platform for managing voting activities while distinguishing user privileges through role-based access.

3.4 ER Diagram

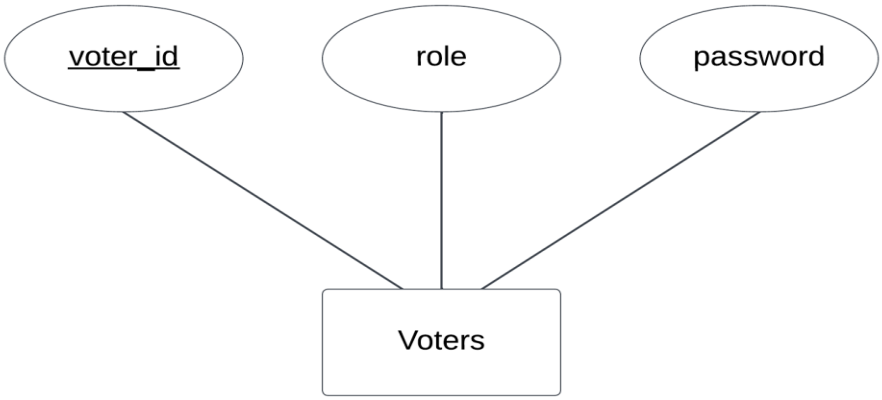

Figure 5 ER Diagram

The figure 5 represents the Voters database schema used in a voting system. The table contains three attributes: voter_id, role, and password. The voter_id serves as the primary key, uniquely identifying each voter in the database. The role attribute indicates the user's role within the system, such as "voter" or "admin," to differentiate access privileges. Lastly, the password attribute stores the voter's login credentials, ensuring secure access to the system. This schema is essential for managing voter authentication and authorizing actions based on the assigned role, forming a secure foundation for the voting system.

3.5 Use Case Diagram

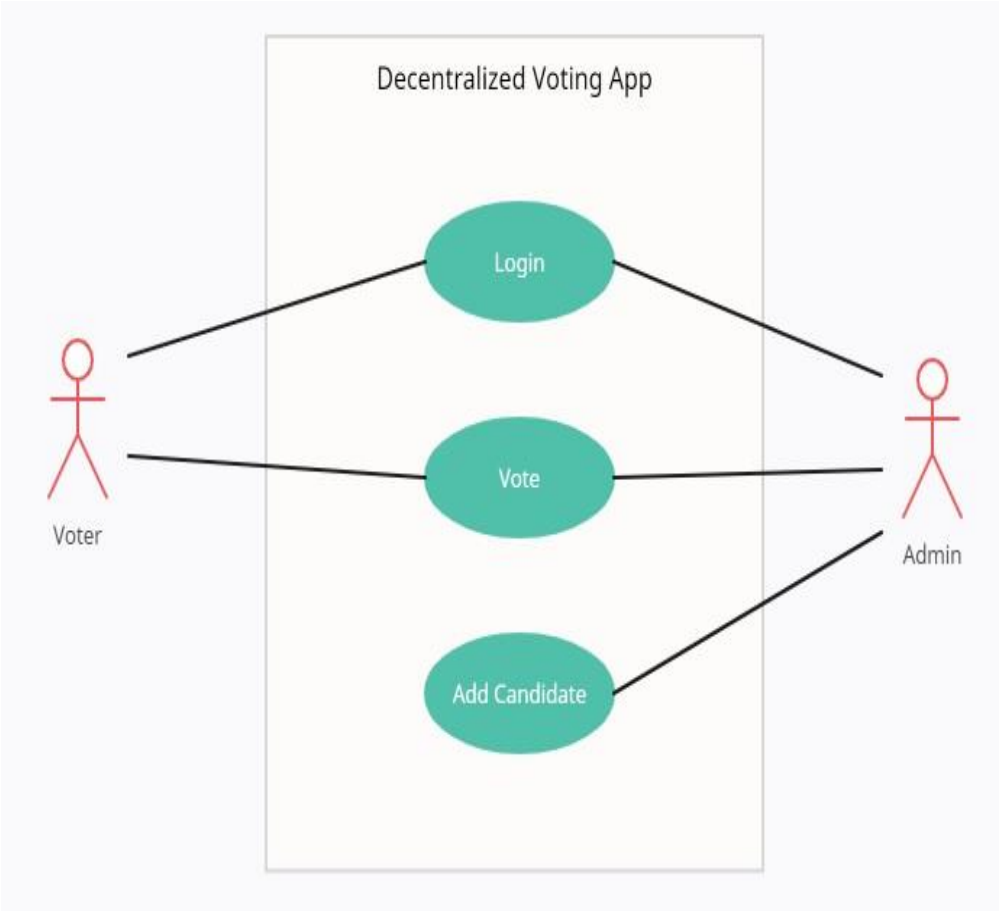

Figure 6 Use Case Diagram

The figure 6 use case diagram illustrates the use case model for a Decentralized Voting App, showcasing the interactions between two primary actors: the Voter and the Admin. Both actors can log in to the system, as depicted by the shared Login use case. Once logged in, the Voter can participate in the voting process through the Vote use case, allowing them to cast their vote for their chosen candidate. On the other hand, the Admin has additional privileges, including the ability to manage the system by using the Add Candidate use case to add new candidates to the election. The diagram visually outlines these functionalities and clearly distinguishes the roles and interactions of each actor, emphasizing the app's decentralized and role-specific design.

4. 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.

4.1 Types of Testing

4.1.1 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.

4.1.2 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.

4.1.3 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.

4.1.4 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.

4.1.4 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.

4.2 Test Results

Table 2: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.

Table 3: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.

Table 4:Test Case3

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.

Table 5: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.

Table 6: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.

5. CONCLUSION AND FUTURE ENHANCEMENT

5.1 Conclusion

The decentralized voting system built on the Ethereum blockchain provides a secure, transparent, and tamper-proof solution for elections. By utilizing blockchain technology, the system ensures the integrity of each vote, preventing tampering or alteration. The system's decentralized nature removes the need for intermediaries, such as election commissions, thus reducing the chances of manipulation and fraud. Additionally, the transparency and immutability of blockchain help in increasing voter trust in the electoral process. With the integration of smart contracts, the entire process, from voter registration to result tallying, is automated and secure, which simplifies the election process and provides real-time results. This decentralized system not only enhances the efficiency and security of elections but also supports a more inclusive and accessible voting process by enabling remote participation, allowing voters to cast their ballots from anywhere in the world.

As the world continues to embrace digital solutions, the decentralized voting system represents a significant leap towards modernizing electoral systems globally. With continued development, such systems have the potential to revolutionize how democratic processes are conducted. Future enhancements, such as improved scalability, advanced voter authentication mechanisms, and the integration of emerging technologies like AI and biometrics, can further bolster the system's effectiveness. These improvements would enhance the system’s accessibility, security, and overall trustworthiness, paving the way for more efficient, inclusive, and fair elections in the future. This blockchain-based solution holds promise in making the democratic process more transparent, accountable, and resilient to manipulation.

5.2 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.

BIBLIOGRAPHY

[1]. Sandhya Avasthi, Khushboo Jain, Ayushi Prakash, Tanushree Sanwal, "A Blockchain Architecture for Secure Decentralized System and Computing", 2024 3rd International conference on Power Electronics and IoT Applications in Renewable Energy and its Control (PARC), pp.415-420, 2024.

[2]. Prasad R. Patil, Dillip Rout, Sagar S. Mohite, "A Survey of\xa0Decentralized Digital Voting System Using Blockchain Technology", Data Science and Applications, vol.820, pp.27, 2024.

[3]. John Amanesi Abubakar, Azeez Oluwatobi, Omolola Faith Ademola, "Advancing Democratic Governance with AIoT-Enabled E-Voting: A Case Study of Covenant Universitys Departmental Associations in Alignment with SDG 16", Artificial Intelligence of Things for Achieving Sustainable Development Goals, vol.192, pp.335, 2024.

[4]. Sumedh V. Chinchamalatpure, Smita R. Kapse, Aditya K. Balki, Vaishnavi A. Rahangdale, Vidisha K. Fulzele, Shital Telrandhe, "Design of Blockchain-based Secured Election Voting System", 2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp.1-7, 2024.

[5]. Garima Verma, Soumen Kanrar, "Secure Keyword Search over Encrypted Cloud Data Using Blockchain in Digital Document Sharing", Wireless Personal Communications, 2024.

[6]. A. Balti, A. Prabhu, S. Shahi, S. Dahifale and V. Maheta, "A Decentralized and Immutable E-Voting System using Blockchain," 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), Coimbatore, India, 2023, pp. 1434-1439, doi: 10.1109/ICSCSS57650.2023.10169552.

[7]. M. V. R. S. Kiran, S. Suchitra, K. Arthi, N. Senthamarai and M. Jayaselvi, "Design and Web Development of EVoting System," 2023 International Conference on Networking and Communications (ICNWC), Chennai, India, 2023, pp. 1-5, doi: 10.1109/ICNWC57852.20 23.10127445.

[8]. H. A. Almahadin, M. Shehadeh, A. S. Al-Gasaymeh, I. A. Abu-AlSondos and A. A. B. Atta, "Impact of Blockchain Technology and Fintech on Sustainable Performance," 2023 International Conference on Business Analytics for Technology and Security (ICBATS), Dubai, United Arab Emirates, 2023, pp. 1-5, doi: 10.1109/ICBATS57792.2023.10111313.

APPENDIX

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);
$("#dates").text( startDate.toDateString(("#DD#/#MM#/#YYYY#")) + " - "
+ endDate.toDateString("#DD#/#MM#/#YYYY#"));
}).catch(function(err){
                console.error("ERROR! " + err.message)
              });
});
for (var i = 0; i < countCandidates; i++ ){
instance.getCandidate(i+1).then(function(data){
var id = data[0];
var name = data[1];
var party = data[2];
var voteCount = data[3];
var viewCandidates = `<tr><td> <input class="form-checkinput"
type="radio" name="candidate" value="${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://trufflframework.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=${voter_id}&password=${pas
sword}`, { 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?Authoriz
ation=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?Authoriz
ation=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",
  "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"
  }
}
```