CLARK CONSULTING™

Software Design Specification

Decentralized Voting Machine

Table of Contents

Revision History

Version	Name	Reason For Changes	Date
1.0			

Approved By

Approvals should be obtained for project manager, and all developers working on the project.

Name	Signature	Department	Date

1. Introduction

1.1 Purpose

This document contains the highlevel requirements for decentralized voting system. It outlines the technical specifications and requirements for the development and implementation of the system. The purpose of this document is to provide a clear and comprehensive description of the system design, including the system architecture, components, and interfaces.

1.2 System Overview

This project incorporates additional fields and features to the existing voting system. This makes the process of voting more secure by making use of blockchain technology. This document contains the architecture, data flow and user interface of the project. This would provide a comprehensive overview of the decentralized voting machine system, including its design, functionality, security, performance, and deployment. This document would serve as a blueprint for the development team to guide the implementation and testing of the system.

1.3 Design Map

I. Introduction
- *Purpose and goals of the system*
- *Benefits of a decentralized voting system*

II. System Architecture
- *Overview of the system architecture*
- *Hardware and software components*
- *Interactions between components*
- *Communication protocols and standards*

III. Data Flow
- *Overview of the data flow within the system*
- *Data input, storage, processing, and output*
- *Data transfer and communication protocols*

IV. User Interface Design
- *Design of the user interface*
- *Graphical user interface (GUI)*
- *Other interfaces that users may interact with*

V. Security
- *Overview of security measures implemented in the system*
- *Encryption of data*
- *Authentication and authorization mechanisms*
- *Other security features*

VI. Conclusion
- *Summary of key points in the system design document*
- *Important considerations and challenges for the project*

1.4 Definitions and Acronyms

Voter – The person who is eligible to cast vote.
Ballot – A system of voting secretly.
Affiliation – The state or relation of being closely associated with a particular person, group, party, company, etc.
Platform – A formal set of principal goals which are supported by a political party or individual candidate, in order to appeal to the general public

2. Design Considerations

2.1 Assumptions

2.1.1 **Availability of a suitable blockchain platform:** The assumption is that there is an available blockchain platform that can be used for the development of the decentralized voting machine. The platform should be scalable, secure and able to handle a large number of transactions.

2.1.2 Availability of necessary resources: The development of a decentralized voting machine using blockchain technology requires expertise I blockchain development, smart contract programming and web development. The assumption is that there are enough resources available to handle the development process.

2.1.3 **Trust in the blockchain platform:** The assumption is that there is a high level of trust in the blockchain platform used for the development of a decentralized voting machine. This is important since the security and reliability of the system depends on the trustworthiness of the blockchain platform.

2.1.4 **Acceptance off blockchain technology:** The assumption is that there is an acceptance of blockchain technology as a reliable and secure means of conducting elections. The adoption of the technology may face some resistance due to concerns about its complexity, lack of familiarity and other factors.

2.1.5 Regulatory compliance: The assumption is that the decentralized voting machine will comply with the relevant regulations and laws governing the conduct of elections in the particular jurisdiction where it is deployed. This includes issues such as privacy, data protection and transparency.

2.2 Constraints

2.2.1 Security: One of the most important constraints is ensuring the security of the voting machine. The blockchain technology used must be secure and robust enough to prevent any unauthorized access or tampering with the voting records.

2.2.2 Accessibility: The voting machine must be accessible to everyone, regardless of their technical knowledge or ability to use technology. This means that the user interface must be easy to understand and navigate.

2.2.3 Scalability: The voting machine must be able to handle a large number of voters simultaneously. This requires a scalable blockchain architecture that can handle high transaction volumes without compromising on performance.

2.2.4 Privacy: The voting machine must ensure the privacy and anonymity of the voters. This requires the use of cryptographic techniques to encrypt the voting records and prevent any unauthorized access.

2.2.5 **Transparency:** The voting machine must be transparent, so that all voters can verify that their vote was recorded correctly. This requires a publicly accessible blockchain ledger that allows anyone to verify the authenticity of the voting records.

2.2.6 **Reliability:** The voting machine must be reliable and trustworthy, so that voters have confidence that their votes will be counted accurately and without any interference.

2.2.7 Regulatory compliance: The voting machine must comply with all relevant regulations, including election laws and data protection regulations.

2.3 System Environment

The Decentralized Voting System can be accessed by the user's server by logging in with their corresponding MetaMask account. The users are made aware about the candidates details one by one. The user's can caste their vote accordingly. Multiple attempts are nullified.

2.4 Design Methodology

The whole project is divided into two phases.

Phase 1: For the front-end application, where the front-end module for interactive user-interface for administrator as well as the user is covered.

Phase 2: For the back-end application, where the back-end module for the implementation of blockchain using Ethereum framework with the help of solidity to convert it into a decentralized system is covered.

2.5 Risks and Volatile Areas

2.5.1 Security risks: Decentralized voting machines are vulnerable to hacking, malware attacks and other forms of cybercrime. This can result in the manipulation or theft of votes, potentially leading to a compromised election result.

2.5.2 **Privacy risks:** Blockchain technology allows for transparent and immutable records, which can be a double-edged sword in a voting system. While it can enhance transparency, t can also lead to the disclosure of voter's identities and voting patterns, potentially undermining the secret ballot principle.

2.5.3 **Technical challenges**: Developing and implementing a blockchain based voting system requires significant technical expertise and resources. The system must be able to handle a large volume of transactions securely and efficiently and must be resilient to network disruptions and other technical issues.

2.5.4 **Legal and regulatory challenges:** The use of decentralized voting machines may not be legal in some jurisdictions or may be subject to complex regulatory frameworks. Compliance with these regulations can add additional complexity and cost to the development and implementation of the system.

2.5.5 **Adoption challenges:** Despite the potential benefits of a decentralized voting machine, there may be resistance from some stakeholders, including political parties ,governments and voters themselves. Convincing stakeholders of the benefits of the system and overcoming their concerns may be significant challenge.

3. Architecture

The architecture provides the top level design view of the decentralized voting system and provides a basis for more detailed design work.

3.1 Overview

This was split into two phases for aiding In the designing and development of the system in a modular and scalable manner. This approach of development was beneficial because:

3.1.1 **Modularity:** Splitting the system into front-end and back-end phases means that each component can be designed and developed separately. Thus making it easier to maintain and update.

3.1.2 Flexibility: Separating the phases so that they can be developed using different technologies. For instance, front-end can be developed using a web or mobile application while the back-end can be developed using blockchain technology and other secure databases.

3.1.3 Team Collaboration: Team members with different skill sets such as designers, developers and so on can work on each phase independently leading to better results and faster development.

3.2 Admin Module

Divided into five components

3.2.1 Dashboard: Contains the details regarding the umber of parties, number of voters, etc**.**

3.2.2 **Add Candidate:** The admin can add candidates standing for the election. After the addition, they are displayed on the user side.

3.2.3 **Create Election:** Allows admin to create election. The user can caste their vote only after the election is created. User can caste vote between the start and end date.

3.2.4 **Election Details:** Admin can update election details like start date, end date, etc.

3.2.5 **Candidate Details:** All the candidates added by the admin are displayed. Admin can update the details of a candidate in case of error.

3.3 User Module

3.3.1 **Dashboard:** Contains information about parties and their candidates.

3.3.2 Voter Register: User has to firstly register themselves. Then they are eligible to caste vote.

3.3.3 Voting Area: After registration only, the user's will be directed to this page, where they can cast their vote.

3.3.4 **Results:** Able to see the results of the election.

3.4 Strategy for Admin as well as User Modules

Technologies such as React and Next.JS are used for the development of interface modules. React is used for building interactive user interfaces and web applications quickly and efficiently with significantly less code when compared to its counterpart Vanilla JavaScript. Using React, applications are developed by creating reusable components that are similar to independent Lego blocks. NextJS is a react framework that gives us building blocks to create web applications. This handles the tooling and configuration needed for React and provides additional structures and features to optimize the application.

3.5 Back-end Module

For implementing the Blockchain technology as well as to store the information of the files. For making the system decentralized.

3.6 Strategy for Back-End Module

The technologies included are

3.6.1 Web3.0: It incorporates the concepts of decentralization, blockchain technologies and token-based economics. Hence it puts the power in the hands of the individuals rather than corporations.

3.6.2 NodeJS: To define all the functionality of data, how the smart contract interacts through the application.

3.6.3 Solidity: Used to create smart contracts that implement business logic and generate a chain of transaction records in the blockchain system. A tool for creating machine-level code and compiling it in the Ethereum Virtual Machine (EVM).

3.6.4 Hardhat: Solidity development environment. Consists of components for editing, compiling, debugging and deploying the smart contracts and dApps.

3.6.5 **MetaMask:** Non-custodial Ethereum-based wallet which allows the users to store, buy, convert and swap crypto tokens. Allows access to their Ethereum wallets through a browser extension.

3.6.6 **CSS:** Used for describing the presentation of webpages including the colors, layouts, fonts, etc.

4. Database Schema

4.1 Tables, Fields and Relationships

In a decentralized voting machine, the database can be organized using tables, fields, and relationships. Here's an example of how this might look:

1.Table: Voters Fields:

- *Voter ID (unique identifier for each voter)*
- *Name*
- *Address*
- *Email*
- *Phone number*
- *Public Key (used for cryptographic signatures)*
- *Table: Candidates Fields:*
- *Candidate ID (unique identifier for each candidate)*
- *Name*
- *Party affiliation*
- *Platform*

2.Table: Ballots Fields:

- *Ballot ID (unique identifier for each ballot)*
- *Voter ID (foreign key linking to Voters table)*
- *Timestamp*
- *Signature (cryptographic signature of the ballot)*
- *Encrypted vote (encrypted version of the vote)*

3.Table: Election Fields:

- *Election ID (unique identifier for each election)*
- *Title*
- *Description*
- *Start time*
- *End time*
- *Status (e.g. ongoing, closed)*

4.Table: Votes Fields:

- *Vote ID (unique identifier for each vote)*
- *Ballot ID (foreign key linking to Ballots table)*
- *Candidate ID (foreign key linking to Candidates table)*

The relationships between these tables would be as follows:

- *Each Ballot belongs to one Voter*
- *Each Ballot contains one or more Votes*
- *Each Vote is for one Candidate*
- *Each Election has multiple Ballots*

4.1.1 Databases

BPData and BPDataPCMSTest are likely names for databases used in a decentralized voting machine system for production and testing/development respectively.
Here's how these databases might be used:

1.BPData - Production Database: This database would store the actual data for the decentralized voting machine system during production use. It would likely contain tables for storing information about voters, candidates, ballots, and elections, as well as other relevant data such as user authentication information and cryptographic keys. This database would be accessed by the voting system software during an actual election, and any changes made to it would have real-world consequences.

2.BPDataPCMSTest - Development and Testing Database: This database would be used during the development and testing phases of the voting system software. It would contain a copy of the data from the production database, but with modifications that allow developers and testers to safely test

the software without affecting real-world data. For example, the database might contain test voter and candidate information, as well as pre-filled ballots for testing purposes. Changes made to this database would not affect the production database.
Overall, having separate databases for production and testing/development is a common practice in software development, as it allows for safer testing without risking real-world consequences.

4.1.2 New Tables

Here are some additional tables that could be used in a decentralized voting machine system:

1.Table: Authorities Fields:

- *Authority ID (unique identifier for each authority)*
- *Name*
- *Email*
- *Phone number*
- *Public Key (used for cryptographic signatures)*

2.Table: Proposals Fields:

- *Proposal ID (unique identifier for each proposal)*
- *Title*
- *Description*
- *Start time*
- *End time*
- *Status (e.g. open, closed)*

3.Table: Vote Cast Fields:

- *Vote Cast ID (unique identifier for each cast vote)*
- *Ballot ID (foreign key linking to Ballots table)*
- *Proposal ID (foreign key linking to Proposals table)*

3.Table: Results Fields:

- *Result ID (unique identifier for each result)*
- *Proposal ID (foreign key linking to Proposals table)*
- *Candidate ID (foreign key linking to Candidates table)*
- *Vote Count*

4.Table: Audit Log Fields:

- *Log ID (unique identifier for each log entry)*
- *Timestamp*
- *Action (e.g. user login, vote cast)*
- *User ID (foreign key linking to Voters or Authorities table)*
- *Description (additional information about the action)*

The Authorities table would store information about the entities responsible for overseeing the election, such as government agencies or election commissions. The Proposals table would store *information about any proposals being voted on, such as referendums or amendments to the voting system itself. The Votes_Cast table would store information about the votes cast on each proposal. The Results table would store the final vote counts for each proposal. The Audit_Log table would store a record of all actions taken within the system, for the purposes of transparency and accountability.*

These additional tables would provide a more comprehensive view of the election process, allowing for greater transparency and accountability.

4.1.3 New Fields(s)

Here are some additional fields that could be added to the existing tables in a decentralized voting machine system:

Table Name	Field Name	Data Type	Allow Nulls	Field Description
Voters Table	*Date of Birth*	*Varchar(50)*		*This field could be used to verify a voter's eligibility to vote based on their age.*
	Vote status	*Varchar(50)*		*This field could be used to indicate whether a voter is registered, eligible to vote, or has already cast their ballot.*
Candidates Table	*Image*	*IMAGE*		*This field could store a picture of the candidate for identification purposes.*
Ballots Table	*Location*	*Varchar(50)*		*This field could be used to indicate where the ballot was cast (e.g. polling station, online voting platform).*
Election Table	*Type*	*Varchar(50)*		*This field could indicate the type of election being held (e.g. primary, general, special).*
Votes Table	*Weighted vote*	*Varchar(50)*		*This field could be used in situations where certain voters have more voting power than others, such as in shareholder or board member elections.*

5. High Level Design

1. Overview

The decentralized voting machine is designed to allow citizens to vote securely and anonymously. The system is decentralized, meaning that it does not rely on a central authority to verify votes. Instead, it uses blockchain technology to ensure the integrity of the voting process.

2. Architecture

The decentralized voting machine will consist of the following components:

- *User Interface: This is the interface that the voter will interact with. It will be a web-based interface that can be accessed from any device with an internet connection.*
- *Backend: This is the part of the system that will manage the voting process. It will be a decentralized application (dApp) that runs on a blockchain.*
- *Blockchain: This is the decentralized ledger that will be used to store the votes. It will be a public blockchain that is accessible to all participants in the network.*

3. Design

The design of the decentralized voting machine will be based on the following principles:

Anonymity: The voting process will be completely anonymous. Voters will not be required to provide any personally identifiable information.

Transparency: The voting process will be transparent, meaning that anyone can view the results of the election.

Security: The voting process will be secure, meaning that it will be impossible to tamper with the results of the election.

Accessibility: The voting process will be accessible, meaning that anyone with an internet connection can participate in the election.

4. Implementation

The decentralized voting machine will be implemented using the following technologies:

- *Blockchain: The system will use a public blockchain to store the votes.*
- *Smart Contracts: The system will use smart contracts to manage the voting process.*
- *Web Technologies: The system will use web technologies to create the user interface.*

5. Testing

The testing of the decentralized voting machine will include the following:

- *Functional testing: This will ensure that the system functions as expected.*
- *Security testing: This will ensure that the system is secure and that the voting process cannot be tampered with.*
- *Performance testing: This will ensure that the system can handle a large number of voters.*

6. Maintenance

The maintenance of the decentralized voting machine will include the following:

- *Updating the smart contracts to ensure that they are up-to-date and secure.*
- *Updating the user interface to ensure that it is user-friendly and accessible.*
- *Monitoring the blockchain to ensure that it is functioning as expected.*

6. Low Level Design

This section provides low-level design descriptions that directly support construction of modules. Normally this section would be split into separate documents for different areas of the design.

6.1 Home Page

6.1.1 Connect Wallet

Allows to connect metamask wallet to the website

6.1.2 Drop Down menu

Consists of wallet address, register voter, register candidate and voter list buttons. The voter registration button from drop down menu will take it to next page where the details of voter can be entered. Similarly for register candidate allows to enter the details which after verification the admin can reject or accept.

6.1.3 Timestamp and count

It shows number of voters and number of candidate registered and the current time on the home page

6.2 Workflow sub-processes

From home page, after the connecting the wallet, the user can register himself as a candidate or voter. After giving in the essential details, the admin can further verify it and can accept or reject it.

The candidate registered and approved by admin will be shown in the home page.

After the successful registration voter , the voter can cast a vote for any of the candidate shown in the home page. After the casting of a message labeled ‘already voted’ will be shown to the Voter and is refrained from casting another

7. User Interface Design

7.1 Application Controls

The navigation bar consist of a drop down menu which further Ccnsists of wallet address, register voter, register candidate and voter list buttons. The voter registration button from drop down menu will take it to next page where the details of voter can be entered. Similarly for register candidate allows to enter the details which after verification the admin can reject or accept.