Software Requirements Specification

for

Blockchain Voting System

**Version 1.0 approved**

**Prepared by Mykyta Medvediev,**

**Mariana Mechyk,**

**Ruslan Dzhubuev**

**Riga Technical University**

**02.03.2025**

**Table of Contents**

[**1. Introduction 1**](#_1fob9te)

[1.1 Purpose 1](#_3znysh7)

[1.2 Document Conventions 1](#)

[1.3 Intended Audience and Reading Suggestions 1](#)

[1.4 Project Scope 1](#)

[1.5 References 1](#)

[**2. Overall Description 2**](#)

[2.1 Product Perspective 2](#_2s8eyo1)

[2.2 Product Features 2](#)

[2.3 User Classes and Characteristics 2](#)

[2.4 Operating Environment 3](#)

[2.5 Design and Implementation Constraints 3](#)

[2.6 User Documentation 4](#)

[2.7 Assumptions and Dependencies 4](#)

[**3. System Features 4**](#)

[3.1 Smart contracts initialization 4](#)

[3.2 Voters identification 5](#)

[3.3 Voting phase 6](#_adl1vzq6u9t1)

[3.4 Counting the votes 6](#_ed9lh38qs337)

[3.5 Announcement of election results 7](#_kmmzprjfb4wy)

[**4. External Interface Requirements 8**](#_3j2qqm3)

[4.1 User Interfaces 8](#_1y810tw)

[4.2 Hardware Interfaces 9](#)

[4.3 Software Interfaces 9](#)

[4.4 Communications Interfaces 10](#)

[**5. Other Nonfunctional Requirements 11**](#)

[5.1 Performance Requirements 11](#_2bn6wsx)

[5.2 Safety Requirements 11](#)

[5.3 Security Requirements 11](#)

[5.4 Software Quality Attributes 12](#)

[**6. Other Requirements 12**](#)

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| Initial document creation | 02.03.2025 | Initial document creation | 1.0.0 |
|  |  |  |  |

# Introduction

## Purpose

The Blockchain Voting System's objective is to provide a voting process that is secure, transparent, and immutable by using blockchain technology, while also ensuring voter anonymity, preventing fraud, and allowing for verifiable election results.

## Document Conventions

* Requirements are categorized as high-level (HL) and detailed (DL).
* Important terms are bolded, while technical terms are italicized.
* Every requirement is assigned a unique identifier (REQ-01).

## Intended Audience and Reading Suggestions

Developers: To design and implement the system.

Project Managers: To oversee development timelines and compliance.

Security Experts: To assess vulnerabilities and suggest improvements.

Election Officials and Authorities: To ensure the system meets legal and operational needs.

Voters: To understand security and usability aspects.

## Project Scope

This system is designed to provide secure and fraud-resistant elections, enable anonymous yet verifiable votes, use blockchain’s immutability to prevent tampering, and offer a decentralized, trustless voting platform. The system supports government elections.

## References

*<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>*

1. "Blockchain-Based Electronic Voting System: Significance and Requirements"
   * Authors: Said El Kafhali
   * Published in: Hindawi / Wiley Online Library
   * DOI: [10.1155/2024/5591147](https://onlinelibrary.wiley.com/doi/10.1155/2024/5591147)
   * Publication Year: 2024
   * Source: Wiley Online Library

This document serves as a reference for blockchain-based voting systems, outlining key challenges, system requirements, and potential design solutions for secure and transparent elections.

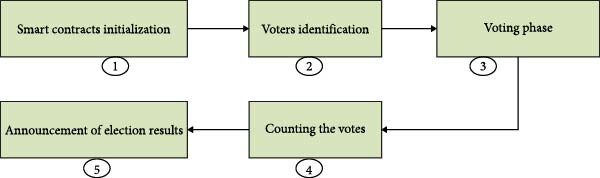
# Overall Description

## Product Perspective

The Blockchain Voting System is a new, self-contained product designed to replace traditional voting methods

## Product Features

The overall voting process contains five essential phases which are presented as follows:

**

## User Classes and Characteristics

The system is designed for multiple user classes, each with distinct roles and requirements:

**Voters:**

* Users; cast votes securely.
* Need minimal technical expertise.
* Require a simple, intuitive interface.

**Election Administrators:**

* Oversee election setup and configuration.
* Ensure voter eligibility and system integrity.
* Require moderate technical knowledge.

**Observers:**

* Monitor the voting process for transparency.
* Use blockchain tools to verify results.

**Developers & Security Experts:**

* Maintain and improve the system.
* Require advanced technical knowledge in blockchain, cryptography, and smart contracts.

## Operating Environment

The Blockchain Voting System operates in a distributed, internet-connected environment with specific hardware and software requirements.

Hardware requirements for a start include desktops and laptops with sufficient blockchain node storage for full-node participants. A stable internet connection is necessary for real-time voting and verification.

Software requirements include a blockchain network such as Local Ganache. Smart contracts are written in Solidity. They will be tested and deployed using Ganache.

The frontend consists of a web application using React.js. With React, users will authenticate themselves using the Metamask wallet.

The system is compatible with Windows, macOS, Linux operating systems.

## Design and Implementation Constraints

Regulatory compliance requires adherence to election laws and cybersecurity regulations such as GDPR and CCPA. Security constraints mandate strong cryptographic mechanisms to ensure vote integrity and voter privacy, as well as protection against Sybil attacks, DDoS attacks, and unauthorized access.

Blockchain limitations include transaction speed and cost depending on the chosen platform. Scalability solutions such as Layer 2 or sidechains may be required to accommodate high voter turnout.

User experience constraints prioritize accessibility and usability to ensure a seamless experience for non-technical voters.

From main tools:

* Visual Studio Code for handling project
* GitHub for code version control
* Solidity is going to be used for smart contracts
* Metamask will be used for authorization
* React.js for building interactive user interface
* Truffle for smart contract compiling, automated testing, and deployment scripts
* Ganache is used for local development that will provide local Ethereum network
* Web3.js for UI and smart contract communication
* PostgreSQL - MOCK database that serves as a storage of eligible voters.
* Node.js - backend for managing eligible voters from the database.

## User Documentation

User manuals provide step-by-step instructions for voters, administrators, and auditors. Online help and tutorials offer interactive guidance within the voting platform. Technical documentation includes API references and developer guides for system integration. Compliance and security guidelines outline best practices for conducting secure and legally compliant elections.

## Assumptions and Dependencies

The voting system requires users to connect a Web3 wallet such as MetaMask to verify their identity and sign transactions when casting a vote. MetaMask extension is used for browsers. This extension should be stable and provide necessary information..

Assumptions include voter access to digital devices and internet connectivity. Election authorities are expected to provide secure voter identification methods. The blockchain network must remain stable and operational throughout the election process.

Dependencies include a blockchain Ethereum platform for storing and verifying votes. Smart contract execution ensures secure and automated vote tallying. Cryptographic identity verification services authenticate voters. MetaMask will facilitate blockchain interactions.

# System Features

## Smart contracts initialization

3.1.1 Description and Priority

The smart contracts are created and initialized with the list of candidates. This list is going to

include only citizens who are allowed to vote. all the voting rules initially established. Any

future modification should be made as maintained by these initial smart contracts*.*

3.1.2 Stimulus/Response Sequences

**Election Authority initializes the election**

* System deploys smart contracts with candidate and voter lists.
* System confirms contract deployment and records initialization.

**System detects unauthorized modification attempts**

* Smart contract rejects modifications outside predefined rules.
* System logs an alert for audit purposes.

**Election starts and voting begins**

* Smart contract validates voter eligibility.
* System records and encrypts the vote on the blockchain.

3.1.3 Functional Requirements

**REQ-1**: The system has to deploy a smart contract that includes candidate information, voter eligibility, and election rules before voting begins.

**REQ-2**: The system should prevent any unauthorized modification of smart contract data once deployed.

**REQ-3**: The smart contract shall reject any attempt to cast votes outside the election period.

**REQ-4**: The system shall log all interactions with the smart contract for auditing and verification.

## Voters identification

3.2.1 Description and Priority

The identification of voters is necessary to assure that votes are not extorted, sold, or stolen. This is very important to decrease the impact of confidentiality loss and to guarantee that the person voting is who they say they are. The voter can identify himself due to some authentication mechanisms such as SmartID, a scan copy of an ID document (Passport or ID card), phone number, biometric authentication, and validity of credentials (public/private key).

3.2.2 Stimulus/Response Sequences

**Voter attempts to log in**

* System requests authentication (e.g., SmartID).
* If authentication is successful, access is granted; otherwise, the attempt is rejected.

**Voter identity verification**

* System checks voter eligibility against the registered voter list.
* If the voter is eligible, the system grants voting access; otherwise, access is denied.

**Unauthorized access attempt**

* System blocks unauthorized users and logs the attempt.

3.2.3 Functional Requirements

**REQ-1**: The system has to authenticate voters using secure identity verification mechanisms.

**REQ-2**: The system should verify each voter’s eligibility against the pre-registered voter list before allowing access.

**REQ-3**: The system shall prevent duplicate logins and multiple voting attempts from the same identity.

**REQ-4**: The system shall log all voter authentication attempts for auditing purposes.

## Voting phase

3.3.1 Description and Priority

Over this phase, the voter chooses candidates according to the voting rules initially established in the smart contracts. The ballot is then hashed using an Ethereum-specific hash function. After that, the hashed or encrypted ballot is however added to the blockchain.

3.3.2 Stimulus/Response Sequences

**Voter accesses the voting interface**

* System verifies authentication and eligibility.
* If verification is successful, the voter proceeds to the ballot.

**Voter selects a candidate and submits the vote**

* System encrypts the vote and records it on the blockchain.
* System confirms vote submission and provides a receipt.

**Unauthorized or duplicate voting attempt**

* System rejects the attempt and logs the incident*.*

3.3.3 Functional Requirements

**REQ-1**: The system shall allow only authenticated and eligible voters to access the voting interface.

**REQ-2**: The system has to ensure each voter can cast only one vote per election.

**REQ-3**: The system should encrypt and securely record votes on the blockchain.

**REQ-4**: The system will provide a confirmation receipt to the voter after successful vote submission.

**REQ-5**: The system shall log all voting transactions for audit and security purposes.

## Counting the votes

3.4.1 Description and Priority

In this phase, some audits take place to guarantee that no fraud has been committed. To do this, when the close of the election is declared, it becomes not possible to add votes. If the counting of the votes occurs in parallel with the voting phase, it is required that the present count is not observable to anyone to avoid influencing other voters who have not yet voted.

3.4.2 Stimulus/Response Sequences

**Election ends**

* The system locks the voting process and prevents further vote submissions.
* Smart contracts trigger the automatic counting process.

**Vote tallying process starts**

* System retrieves all recorded votes from the blockchain.
* Smart contracts validate vote integrity and aggregate the results.

**Result computation and publication**

* The final count is displayed to all authorized users.
* A verifiable record is stored on the blockchain.

3.4.3 Functional Requirements

**REQ-1**: The system shall automatically trigger vote counting when the election ends.

**REQ-2**: The system have to retrieve and count votes using smart contracts, ensuring accuracy and immutability.

**REQ-3**: The system should prevent vote modification or tampering after submission.

**REQ-4**: The system have to generate and publish election results in a publicly verifiable manner.

**REQ-5**: The system shall store final results on the blockchain for future reference and transparency.

## Announcement of election results

3.5.1 Description and Priority

Finally, the voting results are communicated and made available to all via a confident and secure channel.

3.5.2 Stimulus/Response Sequences

**Election Ends**

* The system automatically locks the voting process.
* Smart contracts trigger the vote tallying process.

**Vote Aggregation Begins**

* The system retrieves all stored votes from the blockchain.
* Smart contracts validate and count votes according to predefined election rules.

**Result Computation and Storage**

* The final count is computed and stored on the blockchain.
* The system generates a cryptographic proof of the vote count.

**Result Publication and Verification**

* The election results are displayed to all authorized users.
* Observers and election authorities can verify results using blockchain audit logs.

3.5.3 Functional Requirements

**REQ-1:** The system will automatically initiate the vote-counting process upon the election's conclusion.

**REQ-2:** The system should retrieve all votes from the blockchain and ensure they remain unaltered.

**REQ-3:** The system will use smart contracts to tally votes and ensure accuracy.

**REQ-4:** The system has to prevent any modification of votes after submission.

# External Interface Requirements

## User Interfaces

The blockchain voting system will have an intuitive user interface (UI) that caters to both voters and election administrators.

**Anonymous Interface**

The initial page is going to provide the voter ability to authorize. In this place users will use identity tools.

**Voter Interface**The screen layout will be simple and easy to navigate with a clear voting option ensuring the voting process is intuitive for the user. Buttons and functions will include a "Vote" button that initiates the voting process, a "Help" button to provide guidance on how to vote, a "Confirm" button to finalize the vote submission, and a "Cancel" button that allows the user to exit the process without voting. Standard functions include help functionality on every page, clear action indicators, and confirmation messages such as "Your vote has been successfully cast". Error messages will display clear, non-technical information when issues arise, such as "Error: You are not authorized to vote". Keyboard shortcuts will ensure users can navigate using standard shortcuts such as ‘Tab’ for navigation, ‘Enter’ for selecting options, and ‘Esc’ for canceling.

**Admin Interface**The admin interface will feature a dashboard view that displays election statistics, voter participation, and the status of the voting process. Analytics features are going to be visible only after the election period to prevent possible propagation. Buttons for the admin will include "Start Voting", "End Voting", "View Results", and "Export Results". Standard functions will include admin login, results export, user management (such as the ability to approve/disapprove voters), and audit logs for security. Admin-facing error messages could include things like "Unauthorized Access Attempt" and "Failed to Update Voter List".

UI specifications should be documented separately and must follow consistent design guidelines for accessibility, mobile compatibility, and usability standards, such as WCAG compliance for visually impaired users.

## Hardware Interfaces

The blockchain voting system will interact with various hardware components. Below are the logical and physical characteristics

**Supported Device Types**The system will support desktops and laptops for voters. It will also require servers (cloud or on-premise) for blockchain nodes and election data storage.

**Communication Protocols**Voters will access the system through a web browser, such as Chrome, Firefox, or Edge.

**Nature of Data and Control Interactions**Data interactions will include sending encrypted vote data from the user’s device to the blockchain. Server interactions will include receiving data from the voters and validating votes via blockchain consensus.

**Blockchain Hardware**The hardware required to run blockchain nodes, such as servers or cloud instances, must support the specific blockchain network's requirements, including Ethereum nodes and consensus algorithms like Proof of Stake or other custom blockchain designs.

## Software Interfaces

The blockchain voting system will need to interface with several software components. Here’s a breakdown of these components

**Operating System**The system will be compatible with common OS environments such as Linux for blockchain nodes, Windows, macOS for voter interfaces.

**Blockchain Technology**The system will use Ethereum or another blockchain platform for transaction validation and immutability. Smart contracts will be implemented to facilitate the voting process and ensure that votes cannot be tampered with.

**Databases**A secure database will be used to track voter registration, voting status, and election results. For blockchain storage, the system will store each vote as a transaction on the blockchain network.

**APIs and Services**The system will use RESTful APIs or GraphQL APIs to interact with the blockchain, such as querying voter data and storing transactions. Web3.js or Ethers.js will be used to interface with Ethereum nodes for transaction signing and contract interaction.

**Data Items**Incoming data will include voter identification, vote choice, and user authentication. Outgoing data will include transaction hashes on the blockchain for vote recording and vote confirmation for the user.

**Shared Data**Voter data, such as registration and validation, must be shared across software components like the voter registration system and blockchain. Results data will be made available to the admin interface for viewing and reporting.

## Communications Interfaces

The blockchain voting system will need to integrate with various communication systems

**Communication Protocols**The system will use HTTP/HTTPS for secure communication between client devices, such as voters, and server endpoints. WebSockets will be used for real-time notifications, such as "Your vote has been successfully cast" or any updates on the voting status. SMTP will be used for email notifications to voters, such as vote confirmation, reminders, and election updates. Web3 will be used for communication with the Ethereum blockchain or other blockchain platforms for sending and validating votes. It will use JSON-RPC (Remote Procedure Call over HTTP/WebSockets) to provide communication between Frontend & Smart Contracts. Request for a vote eligibility between Frontend and Backend will be done with secured RESTful APIs. Behind the scenes, during creation of blockchain communication between smart contracts & Ethereum Network will be done with P2P protocols.

**Message Formatting**JSON will be used for message formatting in the REST API communication between the voting system's frontend and backend. Ethereum transaction data will follow the Ethereum Transaction format for submitting votes.

**Security and Encryption**HTTPS will be enforced for all communications to ensure data integrity and privacy. End-to-End Encryption will be used for vote data to ensure that votes cannot be intercepted or tampered with during transit. Public/Private Key Encryption will be used for user authentication and transaction signing in the blockchain.

**Data Transfer Rates & Synchronization**Low latency is critical, especially for real-time voting status updates. Communication between the blockchain and the backend must ensure consistency across nodes and users.

By adhering to these interfaces and requirements, the blockchain voting system will offer a secure, transparent, and reliable solution for digital elections.

# Other Nonfunctional Requirements

## Performance Requirements

**Scalability**: The system must be able to handle thousands of concurrent votes without degradation in performance. It should be capable of scaling horizontally by adding more nodes to the network as the number of voters increases.

**Latency**: The system must process each vote within 5 seconds to ensure real-time voting experiences. For confirmation of votes on the blockchain, the system must confirm within 30 seconds.

**Transaction Throughput**: The system should support a minimum of 10,000 transactions per second (TPS) to handle large-scale elections efficiently.

**Uptime**: The system must achieve 99.99% uptime, ensuring it remains operational throughout the voting period. Scheduled maintenance should be carefully planned to minimize disruption.

## Safety Requirements

**Data Integrity**: All votes must be securely recorded on the blockchain to prevent any tampering or unauthorized modifications. Any changes in the blockchain data should require consensus from the network and be auditable.

**Backup and Recovery**: Regular backups of the blockchain data should be performed to protect against potential data loss. In case of system failure, there should be a defined recovery process to restore the system to its last known good state.

**User Protection**: Users should be protected from phishing and social engineering attacks, which could potentially redirect their votes.

## Security Requirements

**User Authentication**: Voter authentication should be achieved through government-issued identity verification. Each voter should only be able to cast one vote.

**Data Encryption**: All data, including votes, personal information, and blockchain transaction data, must be encrypted both in transit (using TLS/SSL) and at rest (using AES-256 or better encryption).

**Blockchain Privacy**: While votes will be public on the blockchain for transparency, private data such as voter identification should be stored off-chain in encrypted databases, with only minimal public identifiers on the chain.

**Access Control**: Administrators should be required to have strict access control mechanisms to prevent unauthorized changes to the blockchain. Administrative actions should be logged and auditable.

**Auditability and Transparency**: The system must allow for full auditing of all actions. This includes logging all user activity, vote submissions, and changes to the blockchain. Voting data should be public, but anonymous, ensuring transparency and privacy.

## Software Quality Attributes

**Reliability**: The blockchain voting system must be highly reliable, with the goal of zero downtime during the voting period. This includes a fault-tolerant network design that ensures continued operation even during node failures.

**Availability**: The system should be highly available, supporting 24/7 operations, with a failover mechanism in place to ensure minimal service interruption in case of system or network failures.

**Maintainability**: The system should be built with maintainability in mind, ensuring that developers can update and fix any issues with minimal downtime. Automated testing and continuous integration pipelines should be in place for smooth updates.

**Scalability**: The system must be scalable both horizontally and vertically, able to grow in capacity as needed without significant changes to its architecture. This includes the ability to handle a high number of simultaneous voters during peak periods.

**Security**: As outlined earlier, the system must maintain robust security features, including encryption, authentication, and access control, to prevent fraud and unauthorized access.

**Usability**: The user interface should be simple, intuitive, and accessible, ensuring that voters with limited technical knowledge can easily participate in the election. This includes support for multiple languages and accessibility standards.

# Other Requirements

**Appendix A: Glossary**

* **Blockchain**: A distributed digital ledger that records transactions across multiple computers in a way that ensures the data cannot be altered retroactively.
* **Smart Contract**: Self-executing contracts with the terms of the agreement directly written into lines of code on the blockchain.
* **Voter Authentication**: The process of verifying the identity of a voter before they are allowed to participate in the election.
* **Distributed Ledger**: A database that is spread across multiple locations or participants, ensuring transparency and immutability.
* **Proof of Stake (PoS)**: A consensus algorithm used by blockchain networks where validators are chosen based on the number of cryptocurrency tokens they hold and are willing to "stake" as collateral.
* **API**: Application Programming Interface, a set of rules and protocols that allow different software applications to communicate with each other.
* **GDPR**: General Data Protection Regulation, a regulation in EU law on data protection and privacy.

**Appendix B: Issues List**

**Voter Registration Integration**: Pending decision on how the system will integrate with existing voter registration databases.

**Blockchain Selection**: Final decision on which blockchain platform will be used for recording votes.

**Vote Privacy**: Determining the final approach to maintaining vote privacy while ensuring transparency and auditability.

**Internationalization**: Deciding on the number of languages to support at launch.

**Performance Testing**: Pending performance tests to determine how the system handles a large-scale election with millions of votes.

**Compliance with Local Laws**: Ensuring the system meets the legal and regulatory requirements for each jurisdiction where it will be used.

**Data Backup Protocols**: Resolving the best practices for data backup and recovery in case of blockchain node failure.