UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: E-voting Platform for direct democracy on communities using blockchain

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en Tecnologías de la Información

Autor/a:

Franz Paúl Guzmán Basurto

Tutor/a:

Francesc Antón Castro, Ph.D.

Urcuquí, agosto 2023

Autoría

Yo, **Franz Paúl Guzmán Basurto**, con cédula de identidad **1721888293**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, agosto 2023.

Franz Paúl Guzmán Basurto

CI: 1721888293

Autorización de publicación

Yo, **Franz Paúl Guzmán Basurto**, con cédula de identidad **1721888293**, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urcuquí, agosto 2023

Firmado electrónicamente por:
FRANZ PAUL GUZMAN BASURTO

Franz Paúl Guzmán Basurto

CI: 1721888293

Dedication

I am humbled and grateful for what I have achieved through this thesis project, and I am proud to dedicate it to those who may find it useful, especially to my friends and family who have supported me every step of the way.

I will also dedicate this thesis to those who have been marginalized and excluded from the political process, and to those who have not been given a fair chance to make their voices heard. I hope that this work will inspire and empower individuals and communities to take control of their governance and strive for a more just and equitable society.

Franz Paúl Guzmán Basurto

Acknowledgments

Gratitude fills my heart as I express my sincere thanks to those who believed in me from the very beginning. Without their unwavering support, this journey wouldn't have been possible.

I sincerely appreciate my advisor, Francesc Antón Castro, Ph.D., for his invaluable guidance and support throughout my thesis writing journey. His passion for the topic and his expertise in the field have been evident in the many engaging discussions we have had.

I am deeply grateful to my father, Franz, for his invaluable guidance and support throughout the thesis writing process. His extensive experience in directing theses and his expertise were instrumental in helping me review and improve this work. His insightful comments and suggestions have challenged me to think critically and have greatly enhanced the quality of my thesis.

To my manager, Cristhian, for not only encouraging me to complete this project but also for providing me with insightful discussions and suggestions during our one-on-one meetings. His support has been key for my personal and professional growth. To my family for their support and patience throughout my thesis writing journey. Their encouragement and belief in me have kept me motivated and focused on my goals. Especially my sister, Nahomy, for being an inspiration and role model in my life. And my mother, Janeth, for her unconditional love and care.

I would like to take this opportunity to express my heartfelt gratitude to my girlfriend, María Eugenia, for her unwavering support and encouragement throughout this thesis writing journey. I am grateful to have her by my side and look forward to many more adventures to come.

Franz Paúl Guzmán Basurto

Resumen

Las democracias presentan falencias, en particular el desafío de lograr una representación adecuada sin una rendición de cuentas continua. Esta falta de supervisión lleva a un electorado ignorado a volverse apático hacia el voto, profundizando la desconfianza. Los mecanismos que fomentan la participación ciudadana aumentan el apoyo a causas justas, reconocen situaciones ineficaces y mejoran la conciencia pública.

La tesis propone una plataforma de votación electrónica accesible, segura y descentralizada. Aprovechando las ventajas de la tecnología blockchain, como la criptografía y la transparencia, empodera a los miembros de la comunidad para emitir votos sobre propuestas e imponer su visión en sus representantes. El objetivo central es un modelo funcional que utilice lo último en tecnología. El proceso de desarrollo sigue un modelo iterativo, ofreciendo un sistema que cumpla con el objetivo propuesto. Involucra el desarrollo de contratos inteligentes y una interfaz de usuario, pruebas de software exhaustivas y la implementación en una red blockchain personalizada. Las mejoras futuras incluyen la accesibilidad, la seguridad del usuario y algoritmos de consenso más seguros.

La tesis establece un marco referencial para futuras iniciativas en el ámbito de gobernanza, actuando como una guía para aquellos que aspiran a alcanzar objetivos similares. Resalta la importancia de priorizar el bienestar de las personas y mejorar su calidad de vida mediante el apoyo de la tecnología.

Palabras Clave: votación blockchain, contrato inteligente, red blockchain privada, software de votación, votación electrónica, aplicación descentralizada

Abstract

Democracies have flaws, notably the challenge of achieving proper representation without continuous accountability. This lack of oversight leads to an unheard electorate becoming apathetic to voting, deepening distrust. Mechanisms encouraging citizen involvement increase support for righteous causes, recognize ineffective situations, and enhance awareness. The thesis proposes an accessible, secure, decentralized electronic voting platform. Leveraging blockchain technology's advantages, like cryptography and transparency, it empowers community members to cast votes on proposals and enforce their vision on their representatives. The central objective is a functional model utilizing the latest technologies. The development process follows an iterative model, delivering a system fulfilling the proposed objective. It involves smart contract and user interface development, comprehensive software testing, and deployment on a custom-built blockchain network. Future enhancements include accessibility, user security, and more secure consensus algorithms. The thesis establishes a referential framework for future initiatives in the domain of governance, acting as a guiding roadmap for those aspiring to achieve similar objectives. It highlights the significance of prioritizing people's well-being and elevating their quality of life through technology.

Keywords: blockchain voting, smart contract, private blockchain network, voting software, electronic voting, decentralized application

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The most important aspect of a democracy is not only the ability of citizens to share ideas, opinions, and beliefs but also to have their individual voices expressed by voting (collectively) about their future. Representative democracy has its deficiencies, elected representatives do not always follow the plan that they used to win an election [1]. This is one of the reasons why direct democracy mechanisms exist, they allow people to vote on matters that they care about the most. In contrast with this, direct democracy is a political system in which citizens participate directly in decision-making processes rather than through elected representatives. In a direct democracy, decisions are made by a vote of the entire eligible population, typically through popular referendums or initiatives. This form of governance emphasizes direct citizen involvement and aims to give individuals greater power and control over the political process [2]. Direct democracy mechanisms such as public opinion pooling and signature collection are effective tools for knowing public opinion on some issues. Electronic pooling can be a tool for people to make their voices heard and leverage their power in favor of their community.

However, for electronic pooling to proceed as intended, there needs to be a transparent and secure process in which voters knowingly also keep their privacy. The challenge is finding a solution that prevents illegal manipulation of collected data and achieves the desired transparency in security measures to protect voter privacy and collected results [3].

1.1 Background

In Ecuador, elections for choosing representatives have repeatedly resulted in disappointment due to the prevalence of corrupt politicians who consistently fail to uphold their campaign pledges. This trend is prevalent across democracies globally where representatives prioritize their interests over the citizens during the intervals between elections. During these intervals, citizens can influence public policy through intermediaries such as interest groups, social movements, and clientelistic arrangements.

Democracy, as a form of government, involves regular elections that are conducted fairly and counted accurately, giving citizens the opportunity to choose between alternatives offered by political parties. However, one issue with democracy is that elected representatives may not always align with the voters' interests and may change their plans once in power, leading to disillusionment among the electorate [4].

To address this issue, this thesis suggests implementing an electronic voting platform to empower citizens. This platform enables users to create initiatives and gather support from other members of their community, organization, or neighborhood. When a sufficient number of users have supported an initiative, a referendum on that topic may be held to maintain democratic structures.

Similar mechanisms for direct democracy already exist but are not widely used due to logistical challenges in conducting a referendum for each initiative or collecting signatures. Electronic voting systems are a popular solution for reducing the costs of running polls, however, ensuring the reliability and security of these systems is crucial for gaining voter confidence. Several solutions have been proposed to address this challenge.

This work proposes the use of blockchain technology to decentralize the ballots and make them immutable, whose added benefit is to make it harder to manipulate and change the outcome of an election [5]. This project's scope is to develop and provide a tool for the community to start organizing itself. This project does not attempt to replace traditional voting or make other direct democracy mechanisms obsolete but rather gives the people a tool to have an immediate effect on their communities. It will also help elected representatives have constant surveillance of the preferences of their constituents and represent them better, not just when election time comes.

Finally, this project does not address the central issue of corrupt government officials. Still, it does mitigate the effect of a disenfranchised electorate because it empowers the people to change their environment according to their needs and to have their voices heard. An electorate that mobilizes and voices its opinions is more difficult to deceive than a paralyzed one. Representatives will have to be more cautious about going against the will of their constituents if they want to be reelected.

1.2 Problem statement

Representative democracies and top to bottom organizations with elected officials often struggle to transform collective preferences into public policies successfully. Furthermore, those representatives can be corrupted by external influences, such as money or favors done on promises of support. This problem can be mitigated if we start using more mechanisms of direct democracy that do not rely on the decisions of the elected representatives and allow the electorate to have a say in the decisions that affect them. Direct democracies mechanism such as the proposed platform can be helpful tools to organize people in the comfort of their homes.

Electronic voting systems reduce the cost of running traditional elections for both the government and the people by eliminating the cost of printing and transporting ballots and transporting ballot boxes. And they are also less error-prone in gathering the results of the election.

Therefore, the main challenge can be stated, as: *How can one create a minimum valuable product of a decentralized voting system capable of managing initiatives, creating and displaying vote results, and being accurate while employing blockchain technology?.*

1.3 Objectives

1.3.1 General Objective

The main objective of this project is to develop the aforementioned prototype of an electronic voting platform that lets users add proposals and vote on initiatives ensuring transparency, privacy, correctness, and integrity as key pillars.

1.3.2 Specific Objectives

The proposed electronic voting platform must follow the security criteria for electronic voting systems and accomplish the following goals:

1. Program an electronic decentralized voting platform using agile methodologies to create a minimum valuable product.

2. Ensure the integrity, privacy, and reliability of the system using blockchain technology, whose properties provide the solution for these challenges.

3. Guarantee accessibility of the system. By developing a friendly graphic user interface, which delivers an easy-to-use application and facilitates the reading of the results.

4. Mitigate possible system failures. Publishing the web application on a web server ensures that the system is tamper-proof to connection errors.

Chapter 2

Theoretical Framework

The theoretical framework chapter provides a comprehensive overview of the fundamental concepts, tools, and principles required for a thorough understanding of the thesis and its underlying blockchain voting system. It includes an examination of electronic voting systems, the challenges in their development, blockchain technology, and its application in creating decentralized systems, along with the advantages and limitations associated with such systems. This chapter will also cover the necessary tools for decentralized application development, their functionality, and the fundamental principles of software engineering, which are crucial to comprehend the methodology of the project and the functioning of the proposed blockchain voting system.

2.1 Electronic Voting Systems

The section on electronic voting systems provides a comprehensive overview of the technical principles and history of electronic voting systems. It delves into the current state of the technology and sets the standard for what constitutes a suitable prototype for a decentralized voting system. The focus is on enabling members of a community to submit proposals to their representatives for consideration and subsequent action on relevant issues.

2.1.1 History of alternative voting systems

Paper voting is the default choice when running an election. Although paper voting systems are reliable and we have gathered a lot of experience in this subject, they have disadvantages

such as inefficient cost, security, accuracy, accessibility, participation, and sustainability [6]. There have been other options away from the traditional paper voting like punch-card system and mechanical-based. In this section, we will explain the history of alternative voting systems so that we can later get a good understanding of what makes a good voting system.

The first electronic vote recorder was invented by Thomas Edison in 1869. In this system, a signal to a central recorder listed the names of the members in two columns of metal type headed 'Yes' and 'No' [7], and was introduced first in 1886.

From 1849 to 1900, several places around the world started using mechanical machines and lever machines for voting, such as the state of Victoria in Australia or New York in the United States [7]. It will not be until 1901 that we start to see some legal framework fore-voting technologies that ensure adequate protection of human rights. By the end of 1930, almost every major city in the United States was voting using lever machines.

Direct-recording electronic voting systems (DRE), which are physically hardened machines that prevent from the typical PC connectors, were first used in 1986. Another alternative to traditional voting was the Punch Card Voting/Tabulation System. And the optical scan voting system is used together with optometric authentication to read the marked paper ballots and count the results [7].

The telephone network facilitated remote voting, which has provided an alternative voting procedure for a specific subset of the electorate - usually, those with disabilities - in a small but significant number of democratic elections [8].

With the arrival of the Internet, electronic voting became more popular. Its use was widespread across the European Union, in fact, Estonia was the first country to introduce electronic voting over the web, and it fully covered Internet voting including mobile voting [7]. However, internet voting is not widely used today because of its potential risks of fraud and election tampering.

Then came an important technology, which is the basis for this thesis, *blockchain*. It was first conceived as a way to remove the need for central control of a digital currency, properties mainly increased trust, security, transparency, and the traceability of data can be helpful in providing an alternative to electronic voting.

2.1.2 Measures to develop an electronic voting system

As was mentioned, there have been alternatives to the traditional voting system but all those systems have something in common. They are meant to be secure, reliable, and accessible to everyone.

Having said that, a *"good"* electronic voting system must satisfy four important criteria [3]:

1. To be reliable and integral, which means not being altered for malfunctions or bad faith actors.

2. To be private so that a malicious actor cannot trace back a vote of a user.

3. To be accessible to any person who wants to be part of the election. An electronic voting system must be easy to use and understand.

4. To be available any time a person wants to cast a vote when there is a running election.

With these preconditions, then the core of an electronic voting system on which we have to vote on a particular topic with possibles outcomes in *Favor* or *Against*, is as easy as to validate that each vote has been successfully accepted in the count of votes and once the voting phase is finished the results have to be available to the public.

2.2 Blockchain

A ledger is a centralized or decentralized digital record-keeping system that tracks transactions and maintains a history of all changes made to the data. Blockchain is a way of implementing a decentralized ledger, blockchain is a new technology that has gained popularity in the recent decade for it is used in finance, the most well-known application of this technology is the crypto-currency Bitcoin [9], however, it is not the only application for a technology that guarantees traceability and immutability of the data. In simple terms, a blockchain is a data chain that consists of a set of data packages (blocks), where the block is made up of multiple transactions (see Fig. 2.1) and can also carry additional data, for example, the counts of votes.

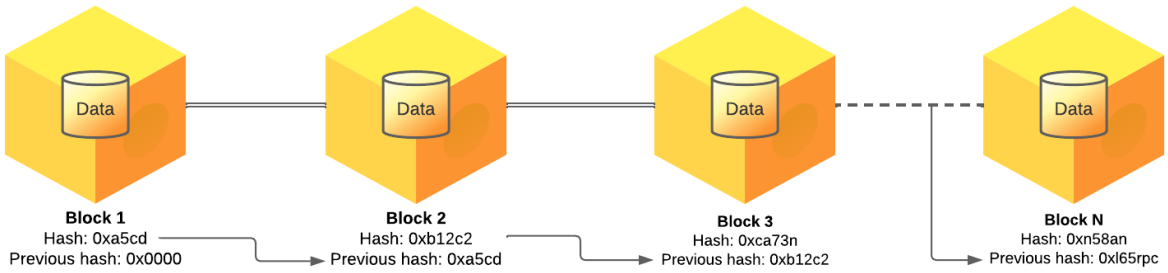


Figure 2.1: A simple blockchain example

The network can validate blocks using cryptographic algorithms. In addition to transactions, each block contains a timestamp and the hash value of the previous block *"parent"*, and a nonce is a random number that helps verify the hash.

This concept ensures the integrity of the entire blockchain through to the first block. Hash values are created by hash functions and there are multiple functions that produce hash values [10], however, hash values have an interesting property, which is that they are a long hexadecimal number intended to uniquely identify each block and its inner content. They help prevent fraud in the block since changes in a block in the chain would immediately change the corresponding hash value.

A node is a participant in the network holding a copy of the ledger (see Fig. 2.2). If the majority of nodes in the network agree with a consensus mechanism on the integrity of block transactions and the validity of the blocks themselves, and if the consensus mechanism is reached, then the block can be added to the chain.

2.2.1 Characteristics of blockchain architecture

Blockchain architecture offers many advantages for various industries that adopt it. It has various built-in characteristics that are outlined below in Table 2.1 [11].

2.2.2 Byzantine General's Problems and consensus mechanisms

How do we ensure that all information is correct before proceeding with an action in a distributed system? A problem called the Byzantine General's Problem [12], can be used to illustrate this concept. Imagine you have a group of generals gathered around a besieged castle with their own armies. Each general must attack at the same time to take the castle

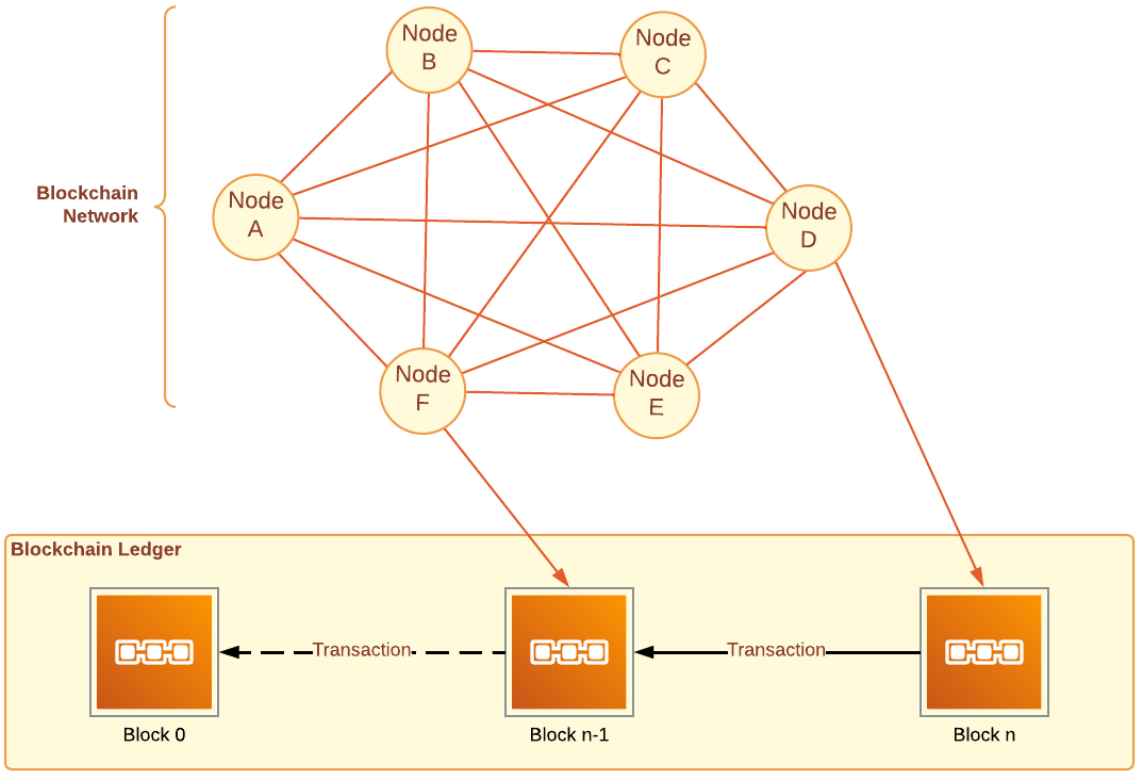


Figure 2.2: An example of a blockchain network

or else they face inevitable failure. Another critical factor in the Byzantine General's problem is that some of the generals are corrupt and untrustworthy; they would send false messages to their commanders so that they would attack at a different time than necessary. These issues can arise in distributed systems because each node must communicate with its neighboring nodes to coordinate its actions. To overcome this problem, all parties involved in a distributed system need to work together when communicating information about an event or change in state.

For instance, what if the enemy captures a messenger and changes the message or if generals do not receive the news? What happens when messengers are spies, changing the content of their messages? What happens if one general is a traitor and sends false information through messengers?

Characteristic	Description
Cryptography	Blockchain transactions are authenticated and accurate due to complex computations and cryptographic evidence between parties involved
Immutability	Blockchain documents cannot be altered or deleted once they have been added to the blockchain
Provenance	Every transaction can be tracked in the blockchain ledger
Decentralization	The entire distributed database is accessible to all members of the blockchain network and a consensus algorithm is used to manage the system
Anonymity	Blockchain network participants use addresses instead of personal identification, which preserves anonymity, particularly in public blockchain systems
Transparency	It is impossible to manipulate the blockchain network as it would require a huge amount of computational resources to change the network

Table 2.1: Characteristics of a blockchain

That is the central issue that blockchain had to overcome to validate the integrity of the message and the messengers. That is the reason why consensus algorithms are used on the blockchain. There are multiple consensus mechanisms and all try to tackle this issue.

In the case of Ethereum, transitioned from a proof-of-work (PoW) based consensus mechanism to a proof-of-stake (PoS) based one. PoW relies on miners who compete to solve mathematical puzzles and create new blocks, while PoS relies on validators who lock up their ether and vote for the correct chain. PoS offers several advantages over PoW, such as lower energy consumption, higher scalability, and stronger security [13].

There are also other types of consensus mechanisms, such as proof-of-authority (PoA), which are used by some permissioned blockchains that require trusted validators [14].

In summary, a consensus mechanism [15] is a process in which a majority (or, in some cases, all) of the network participants agree on the state of a ledger. In other words, it is a set of rules and procedures that allows for maintaining a coherent set of facts between multiple participating nodes.

2.2.3 Types of Blockchain

As the popularity of blockchain technology has grown, so has the number of different types of blockchains. In this table 2.2, we will explore the different types of blockchains, including public, private, and consortium blockchains, and their unique characteristics. Understanding the differences between these types of blockchains is essential to determine which would be suitable for our proposal voting system[16].

2.2.4 Ethereum

After the appearance of Bitcoin, a programmer named Vitalik Buterin saw the potential of blockchain technology and the decentralization feature. That is the reason that he started working on a blockchain protocol that would simulate a quasi-Turing-complete decentralized machine where programs could be executed. In 2015, the Ethereum project was finally released and has since grown in the developer community and has received technical improvements.

As of today, Ethereum stands as a prominent blockchain-based platform for smart contracts, enabling the development and execution of a wide range of decentralized applications (dApps). These dApps leverage the capabilities of smart contracts to facilitate various functionalities such as financial services, supply chain management, decentralized exchanges, governance systems, and more (see Fig. 2.3). The programmable nature of smart contracts on the Ethereum network allows for the automation and transparency of transactions, eliminating the need for intermediaries and enhancing efficiency and trust in decentralized applications, in the following subsections. An explanation of what a smart contract is will be further provided; meanwhile, it is important to note that, unlike Bitcoin, there is no limit for the block size, which will prove useful for the purpose proposed by this thesis. Ethereum is a peer-to-peer network of mutually distrusting nodes that maintain a common view of the global state and execute code on request. The stated is stored in a blockchain secured by a proof-of-work consensus mechanism. [17]. This will also be explained in more detail in the continuous section.

Therefore, the main difference with Bitcoin is that for every transaction in the transaction list, the new state is created by applying the previous state. The block header on the

Type of Blockchain	Description	Advantages	Disadvantages	Examples
Permissionless (Public) Blockchain	A blockchain that is open to anyone without any restrictions on who can use it or participate in the network.	Independence, Transparency, Trust	Performance, Scalability, Security	Bitcoin, Ethereum
Permissioned (Private) Blockchain	A blockchain that is only accessible to authorized individuals or organizations. It operates within a closed ecosystem, where participants require permission to join the network, view the blockchain history or issue transactions.	Access Control, Performance	Trust, Auditability	Hyper Ledger Fabric, Hyper Ledger Besu
Consortium (Federated) Blockchain	A blockchain that is governed by a consortium or group of organizations, that jointly control the network. It is designed to remove power from a single individual or entity, and distribute it among a group of people or organizations.	Access Control, Scalability, Security	Transparency	Quorum, Corda
Hybrid Blockchain	A blockchain that blends essential components of both public blockchain and private blockchain with the mix of the best of both public and private blockchain protocols.	Access Control, Performance, Scalability	Transparency, Upgrading	IBM hybrid blockchain

Table 2.2: Types of blockchain summary

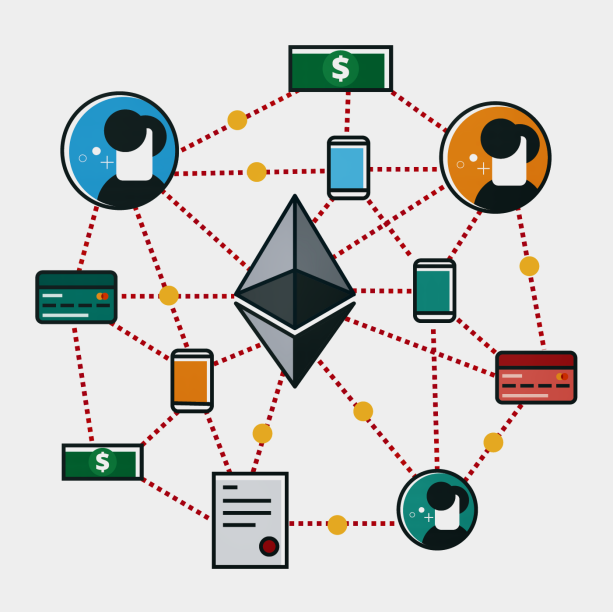


Figure 2.3: Ethereum Network

Ethereum blockchain consists of the Keccak 256-bit hash of the parent block's header, the address of the mining fee recipient, the hashes of the roots of state, the transaction and the receipts tries, the difficulty, the current gas limit of the block, a number representing the total gas used in the block transactions, timestamp, nonce, and several additional hashes for verification purposes [18].

Ether is the currency that this blockchain to perform transactions. Proof-of-Work (PoW) was the consensus algorithm adopted for Ethereum. By successfully solving math puzzles, the winners of this PoW competition are allowed to generate new blocks that contain as many outstanding transactions as possible. As a return, each winner can collect all transaction fees and earn a block reward (if its new block is accepted by other participants). This economic incentive encourages participants to contribute their computation power as much as possible in solving PoW puzzles this is a process called mining in the literature [19].

Specifically, the math puzzle that Ethereum *miners* solve is as follows:

$$\boldsymbol{\sigma}_{t+1} \equiv \Pi\left(\boldsymbol{\sigma}_{\mathrm{t}}, B\right) \tag{2.1}$$

Where $\boldsymbol{\sigma}_{t+1}$ represents the state of the Ethereum virtual machine (EVM) at time $t+1$. Therefore, $\Pi(\boldsymbol{\sigma_t}, B)$ denotes the block-level state transition function, which takes the pre-

vious state $\boldsymbol{\sigma_t}$ and the current block B as inputs to calculate the next state. This function ensures that the execution of transactions within a block is consistent and deterministic, updating the state accordingly.

Let B be a block defined as an ordered sequence of transactions:

$$B \equiv [T_0, T_1, \ldots] \tag{2.2}$$

In this equation, B is represented as a list, denoted by square brackets, containing individual transactions $T_0, T_1, \ldots$. Each transaction is identified by its index in the sequence.

Finally, the right side of equation 2.1:

$$\Pi(\sigma, B) \equiv \Omega\left(B, \Upsilon\left(\Upsilon\left(\sigma, T_0\right), T_1\right) \ldots\right) \tag{2.3}$$

represents the block-level state transition function, where σ denotes the machine state, and Ω refers to the block-finalization state transition function responsible for rewarding a nominated party. It is important to note that the specific implementation details of Ω are not necessary for the reader's understanding as they can vary based on the particular version of Ethereum and the consensus algorithm employed [20].

Ethereum Transactions

Any change in the blockchain state starts with a transaction. This transaction can be directly transferring Ether to another account or could be a trigger for a contract.

A transaction (T) is a single instruction code that sends a message from an Externally Owned Account. Ethereum blockchain launches with a genesis block, and the other transactions process and create a new block and a new state [21].

As Ethereum can be viewed as a transaction-powered state machine, as depicted in equation 2.4 where Υ is the Ethereum state transition function and T represents the transaction.

$$\boldsymbol{\sigma}_{\mathrm{t}+1} \equiv \Upsilon\left(\boldsymbol{\sigma}_{\mathrm{t}}, \mathrm{T}\right) \tag{2.4}$$

The Ethereum state transaction function performs multiple tasks such as checking if the transaction is valid and well-formed, updating nonces and account balances, refunding

the remaining gas, and rewarding miners for the computations.

The Ethereum transaction schema (see Fig. 2.4) defines the structure of a transaction and specifies the various fields and data that must be included in a transaction (see Table 2.3).

Field	Description
Nonce	A unique identifier for the transaction that prevents replay attacks.
Gas Price	The price in Ether that the sender is willing to pay per unit of gas to execute the transaction.
Gas Limit	The maximum amount of gas that the sender is willing to pay for executing the transaction.
To	The recipient address for the transaction.
Value	The amount of Ether or other tokens being transferred.
V, R, S	Components of the digital signature used to verify the authenticity of the transaction.
Data	Additional data to be included in the transaction, such as the vote count.

Table 2.3: Ethereum transaction schema descriptions

When a transaction is sent to the Ethereum network, it must conform to the transaction schema in order to be processed by the network. If any of the required fields are missing or incorrect, the transaction will be rejected by the network.

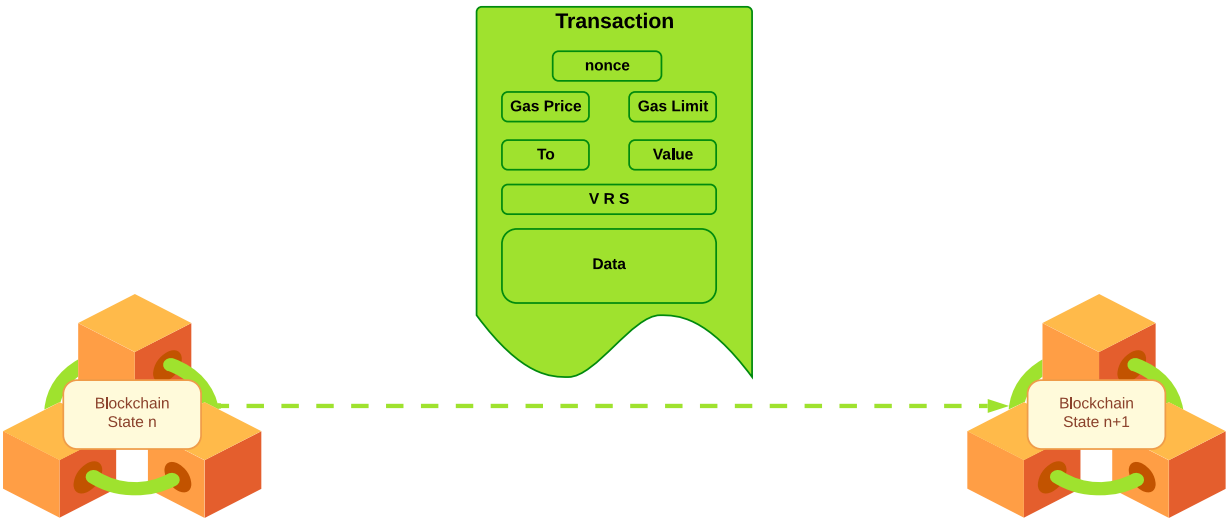


Figure 2.4: Ethereum Transaction Schema

2.2.5 Smart Contracts

A smart contract is a piece of code that serves a unique purpose, like vending machines, the sole purpose of which is to deliver products to its customers and nothing else. In our case, a contract is then a piece of code that will serve the purpose of allowing to vote on initiatives and read the results of the voting.

Smart contracts act as stateful decentralized applications that run on Ethereum Virtual Machine implementations to enforce contract instructions.

Ethereum follows an execution model that specifies how the system state is altered given a series of bytecode instructions and a small tuple of environmental data. This is specified through a formal model of a virtual state machine, known as the Ethereum Virtual Machine (EVM). It is a quasi-Turing complete machine; the quasi-qualification comes from the fact that the computation is intrinsically bounded through a parameter, gas, which limits the total amount of computation done [20].

The EVM is a simple stack-based architecture. The word size of the machine (and thus the size of the stack item) is 256-bit. This was chosen to facilitate the Keccak256 hash scheme and elliptic-curve computations. The memory model is a simple word-addressed byte array. The stack has a maximum size of 1024. The machine also has an independent storage model; this is similar in concept to the memory but rather than a byte array, it is a word-addressable word array. Unlike memory, which is volatile, storage is non-volatile and is maintained as part of the system state. All locations in both storage and memory are well-defined initially as zero [20].

The machine does not follow the standard von Neumann architecture (see Fig. 2.5). Rather than storing program codes in a generally-accessible memory or storage, it is stored separately in a virtual ROM, that can be interacted with only through specialized instructions. The machine can have exceptional execution for several reasons, including stack underflows and invalid instructions. Like the out-of-gas exception, they do not leave state changes intact. Rather, the machine halts immediately and reports the issue to the execution agent (either the transaction processor or, recursively, the spawning execution environment) which will deal with it separately [20].

A smart contract written in the Solidity language undergoes execution within an

Ethereum virtual machine, which then translates the contract into both Application Binary Interface (ABI) and Smart Contract Bytecode. The ABI facilitates interaction with the contract, while the bytecode, together with other parameters, is incorporated into a transaction. After being signed, this transaction is then deployed onto an Ethereum block. The whole process is depicted in Fig. (2.6).

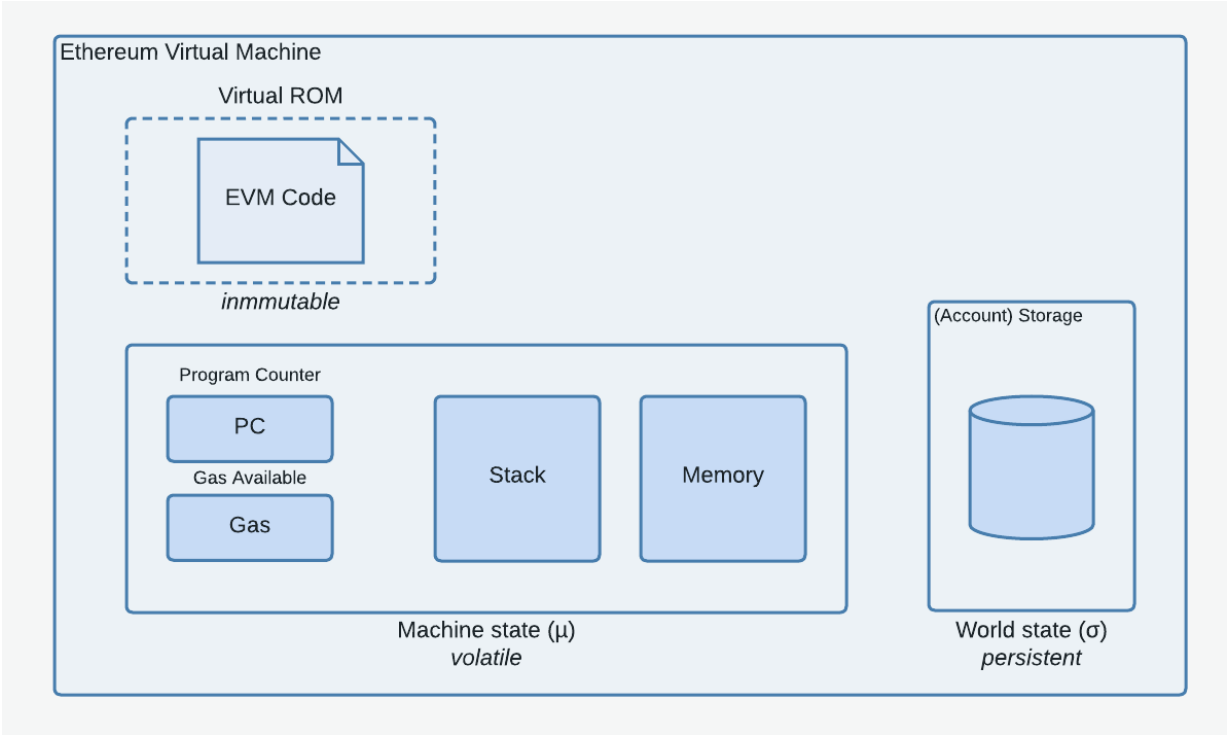


Figure 2.5: Ethereum Virtual Machine

2.2.6 Digital Wallets

Digital wallets in the Ethereum ecosystem are software applications that enable users to interact with their Ethereum accounts. They function akin to online banking applications, providing the ability to view account balances, initiate transactions, and establish connections with various applications [22].

2.2.7 Decentralized Application (dApp)

Ethereum Decentralized Applications can deploy smart contracts to use the capabilities provided by Ethereum to implement business logic. In theory, all processes and data from a

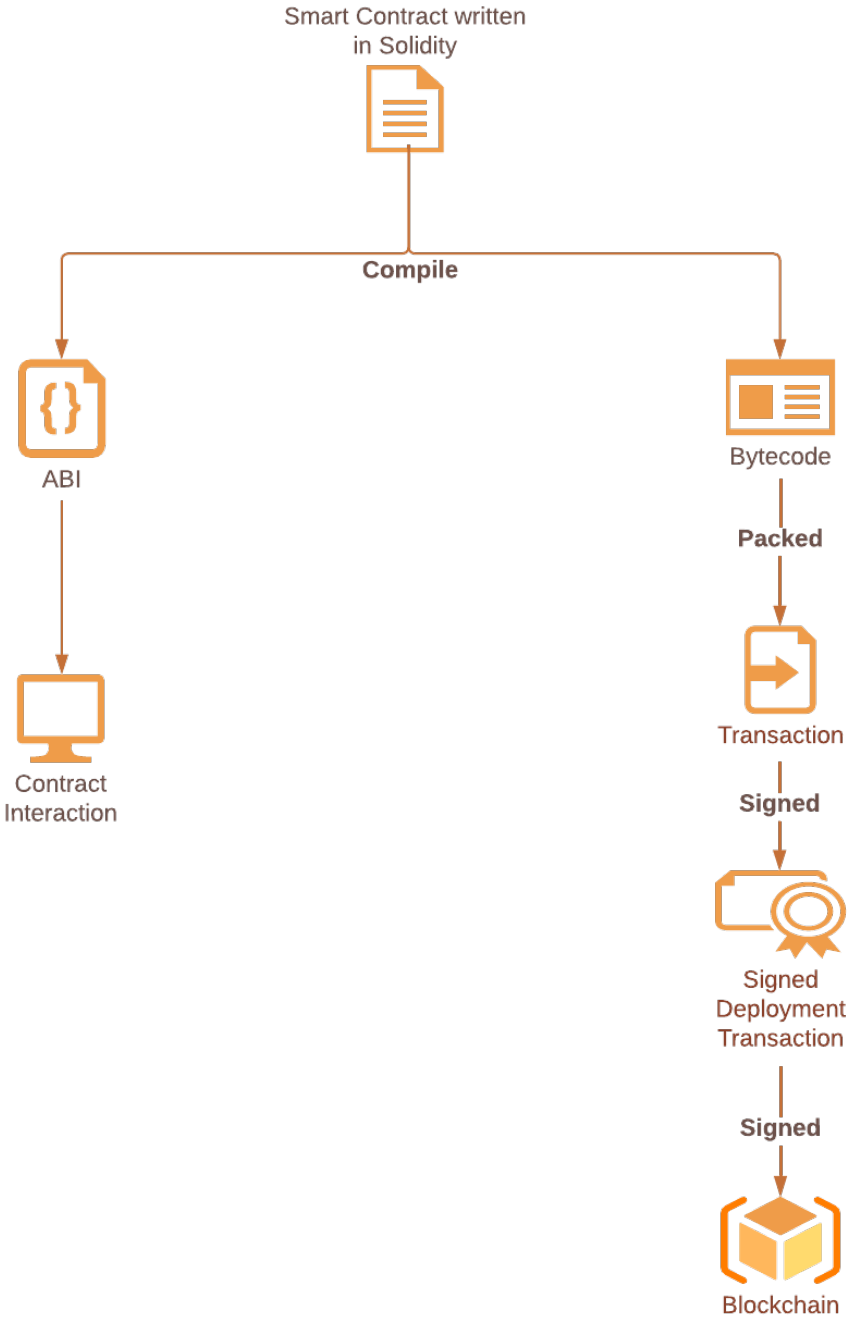


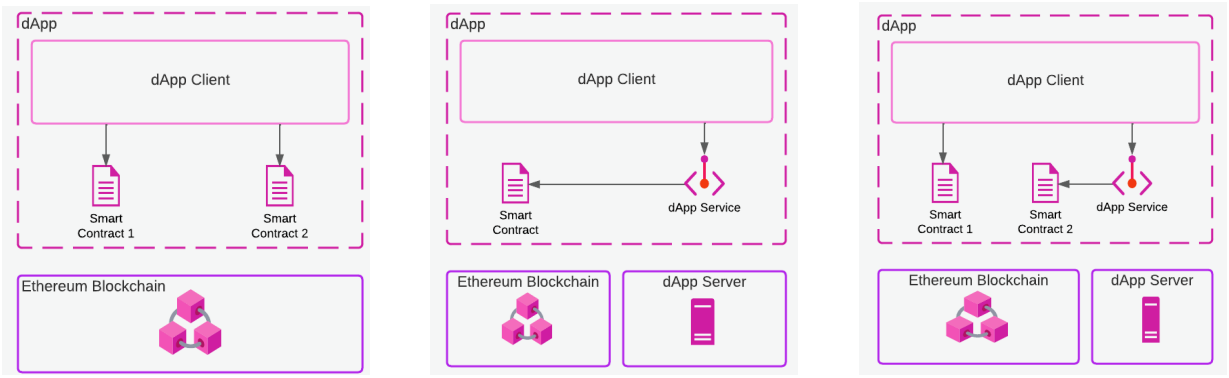
Figure 2.6: Smart Contract Deployment Flowchart

blockchain-based dApp should be handled and stored on the blockchain for pure decentralization. However, due to the performance bottleneck of state-of-the-art blockchain systems, current dApps usually implement only parts of their functionality on the blockchain. As a result, Ethereum dApps adopt three kinds of architecture in practice, as shown in Fig. 2.7: direct, indirect, and mixed. For dApps of the direct architecture (Fig. 2.7a), the client directly interacts with smart contracts deployed on Ethereum. dApps of the indirect architecture (see Fig. 2.7b) have back-end services running on a centralized server, and the client interacts with smart contracts. through the server. Mixed architecture dApps combine the previous two architectures in which the client interacts with smart contracts both directly and indirectly through a back-end server (see Fig. 2.7c) [23]. This thesis proposes a fully decentralized architecture as a voting system is no complex logic that would require an additional back-end service.

Solidity is the programming language for developing smart contracts in the Ethereum community. It is a JavaScript-like language, in which there are contracts (such as classes), functions, and events. The source code of a smart contract is compiled into byte code to be deployed on Ethereum. After the deployment, the smart contract will get an address that allows users to interact with it [23]. All accounts in a network can deploy smart contracts. To do this, accounts must pay for gas for every transaction. The cost of a smart contract consists of two parts: deployment and contract execution. The deployment of a smart contract can be seen as a contract execution of calling a special function constructor. The cost of deployment can represent the complexity of the contract. Since executions are uniformly encoded in transactions and finally packed into blocks, the total gas sent in transactions of a block is limited. Therefore, a contract execution that costs more gas than the limit will fail and will change nothing in the blockchain state [23].

2.3 Software Engineering Concepts

This section delves into the fundamental concepts of software engineering, as it is crucial to understand the best practices and methodologies for software development. The purpose of this section is to lay the foundation for the adoption of agile methodologies in project development, which will help ensure the timely delivery of a software solution that meets



(a) Direct Architecture (b) Indirect Architecture (c) Mixed Architecture

Figure 2.7: Descentralized Application Architectures

the requirements and standards of the software industry.

2.3.1 Agile Methodology

One of the models used in software development is the agile model and it is one of the latest models to be introduced in the software development industry. In comparison with the Waterfall methodology which was inherited from the hardware manufacturers, the agile breed of models focuses on 'agility' and 'adaptability' in development. Instead of one time-consuming and rigid development schedule, agile models involve multiple iterative development schedules that seek to improve the output with every iteration. Each iteration goes through all the steps of design, coding, and testing. The design is not set in stone and is kept open to last-minute changes due to iterative implementation. The team structure is cross-functional, closely knit, and self-organizing. The design idea is never totally frozen or set in stone, but it's allowed to evolve as new ideas come in with each release. Less importance is given to documentation and more to the speed of delivering a working program. Customers may be provided demonstrations at the end of each iteration, and their feedback may determine the next course of changes in the next iteration. The iterative cycle continues until the customer receives a product that exactly meets his expectations [24] (see Fig. 2.8).

The Agile methodology follows a set of rules described in [25], which can be summarized in these key points:

1. The top priority is to please the customer by providing valuable early and continuous

deliveries.

2. Embrace changing requirements, even during later stages of development. Agile procedures leverage changes for the customer's competitive edge.

3. Frequently provide functional software within a couple of weeks to a few months, preferring shorter time frames.

4. Throughout the project, business professionals and developers work together daily.

5. Construct projects around motivated individuals, providing them with the necessary environment and assistance, and relying on them to complete the task.

6. Face-to-face discussions are the most efficient and effective method of exchanging information within a development team.

7. The primary measure of progress is the delivery of functional software.

8. Agile processes promote sustainable development, allowing sponsors, developers, and users to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design improves agility, emphasizing the importance of design quality in maintaining agility.

10. Simplicity is critical, emphasizing the maximization of work not done. Include only what everyone needs, making it easier for teams to add features that address their specific requirements.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. The team regularly reflects on how to become more effective and adjusts its behavior accordingly.

In Agile tasks to be performed are divided into user stories. User stories are a technique to capture and describe software requirements from the perspective of an end-user. The main principle of user stories is to focus on the needs of the user rather than the technical implementation details. They are typically written in a simple, concise, and structured format that includes a title, a description, an estimation in story points, and acceptance

Figure 2.8: Agile Project Life Cycle

criteria. The description should outline what the user wants to achieve, why they want to achieve it, and what value it brings to them. Estimation in story points is an approach to size the effort required to complete a user story. Finally, acceptance criteria define the conditions that must be met for the user story to be considered complete [25].

2.3.2 SCRUM

Scrum is basically an agile, lightweight framework that provides steps to manage and control the software and product development process.

Scrum is the combination of the Iterative model and the Incremental model because the builds are successive and incremental in terms of the features to develop software [26]. Scrum was created to enhance the pace of development, unify individual and organizational goals, establish a performance-driven culture, facilitate shareholder value creation, promote effective communication at all levels, and enhance personal growth and quality of life. While it has gained popularity in recent years and has demonstrated its usefulness, it is

not a methodology that should be employed at all times [27].

Scrum rituals are essential components of the Scrum framework, which is an agile and lightweight methodology designed to effectively manage and control software and product development processes. These rituals consist of specific events that allow teams to collaborate, inspect and adapt their work in an iterative manner.

The key Scrum rituals are depicted in Fig. 2.9. These Scrum rituals foster collaboration, transparency, and continuous improvement within the development team, promoting effective project management and delivering valuable outcomes in an iterative and incremental manner.

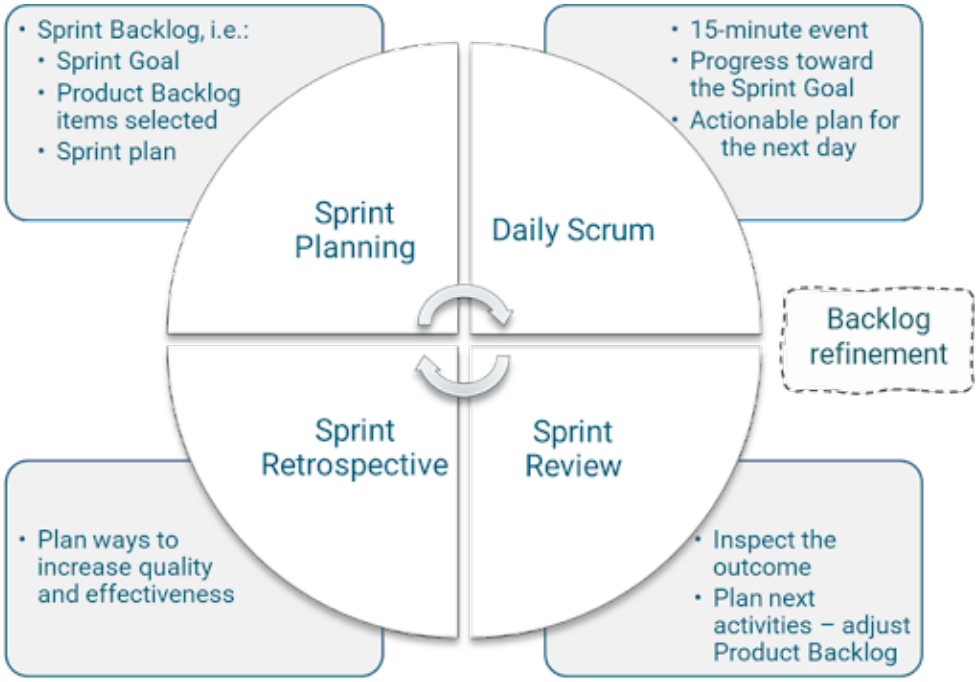


Figure 2.9: Scrum

2.3.3 Minimum Value Product

A Minimum Viable Product (MVP) in Scrum is a development approach where a product is built with just enough features to satisfy early adopters, gather feedback, and validate assumptions about the product's design, market fit, and potential success. The goal of an MVP is to deliver a working product to customers as quickly as possible to learn and iterate based on their feedback [28]. The goal of this thesis is to develop an MVP that satisfies the purpose of allowing a community to add and vote on initiatives that are of

interest to them.

2.3.4 Tech Stacks

Tech stacks refer to the set of software tools and technologies used to build and run a software application. This includes programming languages, databases, web servers, and operating systems, among others. Understanding the components of a software stack and how they work together is important for building efficient and scalable software applications.

The distinction between the frontend and backend pertains to the level of interaction with the developers' code. The frontend encompasses all aspects of the website that are not related to coding and is typically referred to as the public-facing side of the application. This is the portion of the site that users directly interact with [29].

As an administrator, the responsibility of managing and maintaining the website also lies within your jurisdiction. This task cannot be performed from the public-facing aspect of the frontend. A Content Management System (CMS) typically includes an administrative dashboard, known as the backend, that provides functionalities for administrative purposes.

In general, the backend of an application is everything behind the digital curtain, or – in other words – the developers' end. In the case of dApps, it refers to the servers, databases, and other infrastructure components that support the application's functionality and stores its data [30]. Unlike traditional applications, dApps are designed to run on a decentralized network such as blockchain, meaning that the backend is distributed among many different nodes or participants in the network rather than being centralized in a single server or data center. This decentralized architecture helps ensure the availability, security, and scalability of the application, as well as enables users to interact with the application directly without the need for a trusted intermediary.

In this thesis, the backend of the application will be the smart contract that interacts with the decentralized ledger.

Chapter 3

State of the Art

The field of electronic voting has gained significant attention in recent years, with a focus on using blockchain technology to enhance the transparency, security, and integrity of the voting process. This state-of-the-art chapter in a blockchain voting system thesis offers a thorough analysis of current developments and advancements in the field. It summarizes the current understanding of blockchain-based voting systems, highlighting their key attributes, advantages, and limitations, with real-life examples. The chapter also critically evaluates the current state of the art, identifying the main challenges and opportunities for future research. This will serve as a solid foundation for the proposed blockchain voting system that will be presented later in the thesis. The objective of this chapter is to provide a comprehensive understanding of the current state of the art in blockchain voting systems through real-life examples, setting the stage for the design and implementation of a cutting-edge and secure blockchain voting system.

3.1 Decentralized Voting System Proposals

Hanifatunnis (et al.) [31] designed a blockchain-based electronic voting system, that harnessed the power of blockchain to optimally perform voting in a way that avoids collision in the transfer of the results. Also, make sure all nodes that have registered are included in the calculation process. In terms of cost, it can also be more efficient because it does not require equipment, that is always redone in each election carried out.

Hsiao (et al.) [32] proposed the first decentralized voting system on the blockchain, that takes advantage of anonymity and security achieved by the secret sharing scheme

with Paillier's public key cryptosystem and verifiable votes provided by the transparency and non-repudiation of the blockchain.

According to [33], there are two emerging streams of blockchain applications in e-voting. The first one involves the use of blockchain for e-voting (or blockchain-based e-voting, BEV), while the other employs blockchain to support e-voting or voting processes as a third party non-intrusively. This role is synonymous with that of third-party observers in elections. Advocates of Blockchain enabled e-voting (BEV) seek to harness the decentralized blockchain protocols for voting without the control of a central authority such as the electoral management body. BEV attempts to eliminate tampering with votes through cryptographically secure voting records.

Yang (et al.) [34] present an intelligent contract-based, self-tracking, ranked choice decentralized voting system, whose idea is as follows: the election administrator deploys a voting contract by confirming public parameters (such as the public key of the election). Each voter can then submit a vote via the voting contract, with each ballot constituting a transaction of the blockchain system. This system is developed in the Ethereum network.

3.2 Real-life examples of decentralized voting systems

This section showcases real-life examples of successful and ongoing decentralized voting projects, highlighting the practical applications of the technology and the challenges faced during their development and deployment.

By examining these real-life examples, the section provides a comprehensive understanding of the potential and limitations of decentralized voting systems and the practical aspects of their implementation. The objective of this section is to provide a valuable perspective on the current state of decentralized voting systems and to provide a basis for the design and implementation of the proposed blockchain voting system in the thesis.

There have been experiences with decentralized electronic voting systems, according to [35] in Zug, Switzerland. The city proposed using uPort, which is an app that allows users to interact with decentralized applications, as well as identity-related tasks outside the chain through their mobile phones [36]. This app allowed them to provide consulting to the citizens of Zug and broaden their participation, but there are still issues because

there is legal uncertainty about the use of this kind of application.

Another example of blockchain-based electronic voting is described in [37], it is called Agora, a blockchain-powered voting solution intended for governments and organizations, which provides a system from start to finish, that can be checked and validated. Agora has its own token that is utilized on the blockchain for ballots, for which concerned governments and bodies will buy tokens for every individual authorized to cast their vote.

There have been other proposals for using decentralized voting systems such as in Turkey. According to [38], the blockchain-based electronic voting system provides a trusted, secure, and fast voting system for Turkey, and its model could be integrated into other countries with hard work since each country has different laws and election systems.

Other countries are interested in decentralized voting systems as well [39], [40], each molding the technology according to their laws and social context.

Some countries, such as Finland, Estonia, Norway, Switzerland, and others, have been successful in implementing e-voting on a broad scale and increasing legally [41]. The tendency is that more countries will see electronic voting as a solution in the future.

In fact, E-voting in Estonia was established in 2005, and it offers two methods of voting: physically voting at a polling station or e-voting from anywhere in the country. The main goal of the Estonian government was to make the voting process easier and increase voter participation. The traditional voting system requires about 45 minutes to complete, but e-voting only takes 4-6 minutes to cast a vote, reducing the time spent by the voter. E-voting in Estonia is based on Estonian ID cards, which also provide a platform for digital signatures. Voters can choose to vote from home if they have a computer and internet connection, or they can go to public libraries or university labs to cast their votes. There have been proposals to migrate to blockchain voting, as Estonia is also known for promoting the use of blockchain in government, where more than 90 percent of the services are running on blockchain technology and cryptocurrencies [42].

As the adoption of electronic voting systems continues to grow globally, it is likely that more countries will explore the use of blockchain technology to enhance the security, transparency, and integrity of their voting processes. Blockchain's ability to provide a tamper-evident and auditable record of all transactions, as well as its decentralized nature, makes it a promising solution for addressing some of the challenges facing e-voting today

[43].

A proposal to integrate blockchain technology into an existing voting system in the UK was made, allowing citizens to vote at polling places or on the Internet. Netvote is a network utilizing the Ethereum blockchain that uses decentralized applications as part of its user interface. Election administrators can use the Admin dApp to create ballots, set rules, and open or close voting. The Voter dApp allows users to register and cast their ballot, and it can be linked to other devices for increased security. Lastly, the Tally dApp is used to tally and confirm the results of the election as described in [44].

Another real-life success of blockchain voting was performed in the provincial government of Gyeonggi-do, South Korea, in partnership with Blocko, which implemented a blockchain-based voting solution for community funding. This platform allows members of the community and local residents to propose and vote on initiatives for community aid. Over 9,000 votes were received through both online and offline channels, and 527 projects were ultimately selected by the government. According to the government, this blockchain solution provided an opportunity to supplement traditional representative democracy with direct democracy. Blocko, a Korean blockchain research and services start-up developed the CoinStack platform used in this initiative [45].

3.3 Criticism on Blockchain voting systems

This section highlights the security concerns surrounding the technology, scalability issues, usability difficulties, privacy concerns, regulatory uncertainty, and accessibility challenges present in academia. By discussing these criticisms, the section provides a comprehensive examination of the current state of blockchain voting systems and the challenges, that must be addressed in their development and implementation. The goal of this section is to provide a critical evaluation of the technology and to review future research and development efforts in this area.

Jafar [11] highlights that blockchain-based technology is still in its infancy as an electronic voting option and that there are still many technical challenges that need to be addressed. This article discusses recent electronic voting research using blockchain technology. This article identifies and addresses the shortcomings of existing electronic voting

systems. It also explores the potential of blockchain to enhance electronic voting and examines current solutions for blockchain-based electronic voting systems, as well as possible future research paths. Many experts believe that blockchain may be suitable for a decentralized electronic voting system and that it allows all voters and impartial observers to view voting records. However, the research also found that most publications on blockchain-based electronic voting identified and addressed similar issues, and there are still many study gaps in electronic voting, that need to be addressed in future studies. The research concludes that blockchain technologies need more sophisticated software architecture and managerial expertise and that blockchain-based electronic voting systems should be initially implemented in limited pilot areas before being expanded, due to the security concerns that still exist in the internet and polling machines.

Tas and Tanriover [46] provide an overview of the current state of research in blockchain-based e-voting systems. It uses a systematic review approach to analyze 63 research papers found in scientific databases and provides a categorization of the main challenges in the field into 5 categories: general, integrity, coin-based, privacy, and consensus. The main conclusion is that blockchain technology has the potential to provide solutions to problems in current election systems, but there are challenges to be addressed in terms of privacy protection, transaction speed, remote participation security, and scalability. The authors highlight the need for further development and improvement of blockchain frameworks for use in voting systems.

3.4 Improving security on decentralized voting systems

Critics of decentralized voting systems argue that there is no assurance that the voter is who they say they are. Nevertheless, in [47], a biohash to perform authentication is proposed. This biohash is a combination of the individual's fingerprint and ID. Under this model, voters must first register to cast their vote from any mobile device then. Other studies such as in [48] combine blockchain and IoT technology to exchange data from electronic voting machines to the network nodes. While this limits the mobility of blockchain-based voting systems, it significantly enhances security by requiring the voter to physically be

present to cast their ballot.

Another non-remote secure blockchain voting was proposed in [49], the ABVS (Auditable Blockchain Voting System) is divided into three stages: election setup, voting, and counting and verification. During the election setup, trusted public institutions are selected to act as nodes in the ABVS blockchain system, software, and hardware are prepared and certified, unique Vote Identification Tokens (VITs) are generated for voters, and the VITs and ABVS equipment are distributed to polling stations. During the voting stage, voters use the VITs to cast their votes through a secure and encrypted channel to the trusted nodes, where they are verified and added to a vote blockchain. After the election, the counting and verification stage begins, where the election officials deactivate the system, open containers with unused VITs, and send a list of remaining tokens to national electoral central authorities for vote verification. Voters can verify the presence and correctness of their votes by comparing them to the blockchain and the paper trail.

To improve blockchain performance by optimizing the core module, the consensus algorithm [50] uses proofs of assets and proofs of reputation in a voting-based decentralized consensus (VDC) algorithm for consortium blockchain, which is combined with a verifiable random function (VRF). This new algorithm improves the efficiency of the consensus process and user fairness, without compromising on security and energy efficiency.

Another proposal for better consensus algorithms was described in [51], it is an improved version of the delegated proof of stake (DPoS) the new method uses a vague set, which is a mathematical concept that allows for imprecision and uncertainty based on the work of [52] and adapted to human voting. Because in human voting, individuals may not always have a clear preference for a candidate or a decision, and their opinions may be influenced by various factors. For example, they may have some degree of support for a candidate, some degree of opposition, or they may be unsure about their decision. The vague set captures these nuances of human decision-making by using a membership function that includes support degree, against degree, and unsure degree for each candidate or decision. The vague set can be interpreted as a voting model. The membership function of the vague set represents the degree of support or opposition for each candidate and the degree of uncertainty for each vote. The model allows each individual to express their preferences for each candidate or decision more accurately than a simple yes or no vote. This method

allows each node to vote for the agent node, which is a fair and effective way to select the agent node when multiple candidates have the same value. This improves the security and fairness of the blockchain and reduces the chance of malicious nodes being selected as agent nodes. The maximum probability of nodes after the vote of the fuzzy membership degree is 0.5, which means that the voting process is designed to balance the weight of different nodes and prevent any single node from dominating the selection process.

The Open Vote Network is a decentralized, self-tallying internet voting protocol that uses the Blockchain for maximum voter privacy. It is written as a smart contract for Ethereum and can be used for boardroom elections. It does not rely on any trusted authority for tally computation or voter privacy, and each voter has control over the privacy of their own vote. It has been tested on Ethereum's test network and its execution cost has been analyzed to be sufficient by today's standards [53].

Chapter 4

Methodology

This chapter provides a detailed description of the steps taken to design, implement, and operate the system. This chapter aims to provide an overview of the system development process and its contribution to the development of the blockchain voting system, and is further divided into the following sections:

1. Requirements gathering: Explanation of the process of gathering requirements for the system, including any techniques used to gather requirements and how the requirements were validated.

2. Architecture design: Discussion of the design of the system architecture, including the choice of technology and the design of the system components.

3. Introduction to tools for development: Overview of the tools used for the development of the system, including the programming languages, development environments, and other software used to build and test the system.

4. User interface design: Description of the design of the user interface, including any tools used to create the interface and any usability testing performed.

In the following sections, each of these topics will be discussed in greater detail to provide a complete understanding of the system development process and how it contributed to the creation of the blockchain voting system. Through this chapter, readers will gain insights into the development process of a complex system and the challenges that arise when developing a secure and decentralized voting system using blockchain technology.

4.1 Requirements gathering

This section outlines the process of identifying and documenting the functional and non-functional requirements for the proposed blockchain voting system. The section will cover the following sub-topics:

1. Stakeholder Analysis: An identification of all stakeholders who will be impacted by the system, their roles and responsibilities, and their requirements.
2. Functional Requirements: A comprehensive list of the functions and features required in the system, including the necessary inputs, outputs, and data flow.
3. Non-functional Requirements: The quality attributes such as security, scalability, accessibility, and performance requirements that the system must adhere to.
4. User Requirements: The user-specific requirements such as usability, user experience, and accessibility requirements that the system must cater to.
5. Technical Requirements: A description of the technical specifications required to support the functionality of the system, including hardware, software, and network requirements.

This section provides a complete understanding of the requirements that the proposed blockchain voting system must meet in order to be deemed successful. This will serve as the foundation for the design and development of the system and ensure that it meets the needs of all stakeholders involved.

4.1.1 Stakeholder Analysis

Stakeholder analysis is important in an Agile methodology. It helps to identify and prioritize the stakeholders, understand their needs and interests, and determine their level of influence and impact on the system. This information is used to make informed decisions, communicate effectively with stakeholders, and manage their expectations. Stakeholder analysis also helps to ensure that all stakeholders are considered and that the end product meets their needs and requirements [54]. In the context of the decentralized voting system for the creation and voting over initiatives, the stakeholders can be analyzed as follows:

- Members/Users: They are the primary users of the system and play a crucial role in decision-making and voting. It is essential to understand their needs and expectations to design a user-friendly and efficient system.

- Leadership/Governing Body: They have a significant influence on the direction and implementation of the initiatives. It is important to involve them in the requirements-gathering process and understand their expectations to align the system with the organization's objectives.

- Technical Staff: They will be responsible for maintaining and updating the system, and it is important to consider their expertise and limitations in the design and development of the system.

- Regulators/External Auditors: They play a crucial role in ensuring the accuracy and integrity of the voting results. It is important to consider their requirements and regulations in the design and implementation of the system.

- Wider Community/Public: They may be impacted by the initiatives and could potentially be members of the same organization. It is important to understand their expectations and requirements to design a system that meets their needs and ensures their involvement in decision-making.

By conducting a comprehensive stakeholder analysis, this thesis can identify the key stakeholders, understand their interests, and prioritize their needs to design an effective decentralized voting system.

A stakeholder analysis matrix is a tool that allows you to categorize and prioritize stakeholders based on their level of interest, power, and influence in a particular situation or system [55].

In the case of the given stakeholders, a stakeholder analysis matrix is the following Table 4.1.

In this matrix, the columns represent the level of interest, power, and influence of each stakeholder. The rows represent each of the stakeholders.

- Interest: This column reflects the level of interest each stakeholder has in the system,

Stakeholder	Influence	Interest	Impact	Engagement
Members/Users	High	High	High	High
Leadership/Governing Body	High	High	High	High
Technical Staff	Medium	Medium	High	High
Regulators/External Auditors	Low	Medium	Medium	Medium
Wider Community/Public	Low	Low	Low	Medium

Table 4.1: Stakeholder Analysis Matrix

with high interest meaning they have a direct and significant interest in the outcome of the system.

- Impact: This column is used to rate the degree of impact that a particular stakeholder has on the system or system. This column is a subjective measurement of the stakeholder's level of influence or power to affect the outcome of the system. The impact of a stakeholder can be either positive or negative, and it can be measured on a scale of high, medium, or low. This column helps to identify key stakeholders and prioritize their needs, and it helps to ensure that the system meets the needs of the stakeholders who impact its success most.

- Influence: This column reflects the level of influence each stakeholder has over others in the system, with high influence meaning they have the ability to shape the opinions and decisions of others.

- Engagement: This column refers to the level of involvement and interaction that the stakeholder has with the system. It indicates the degree to which a stakeholder is engaged in the development and implementation of the initiative, as well as their level of participation and contribution. The engagement level can range from low, where the stakeholder has minimal involvement or influence, to high, where they have a significant impact on the system and are actively engaged in decision-making and implementation. The engagement level helps to determine the level of resources and attention that should be devoted to each stakeholder and their priority in terms of communication and engagement.

Based on this analysis, it is clear that the Members/Users and Leadership/Governing Body are the most important stakeholders as they have high levels of interest and influence

in the system. Regulators/External Auditors and the Wider Community/Public have a lower level of power but still significantly impact the system's outcome. Technical Staff has a moderate level of interest, and influence over the system.

4.1.2 Functional Requirements

These requirements are a key aspect of the design process and ensure that the system will be able to effectively and efficiently support the voting and decision-making process within an organization. In this section, we will detail the key functionalities and features of the system, along with any constraints or limitations that must be taken into consideration during the development process.

The functional requirements for the system are derived from the stakeholders analysis and the defined objectives. The stakeholders analysis helped identify the different parties interacting with the system and their individual needs and expectations. The objectives set for the system provided the overall direction and purpose for the thesis. The specific actions that the system must perform to meet the objectives and fulfill the needs of the stakeholders. These requirements provide a clear understanding of what the system must do, and form the basis for the design and development of the system as depicted in Table 4.2.

Limitations and constraints in the functional requirements section refer to any restrictions or boundaries that impact the system's ability to meet the specified functional requirements. These may include technological limitations, budget constraints, regulatory limits, performance limitations, and others. Identifying these limitations and conditions early in the requirements-gathering process is essential so that the design and development teams can consider them when building the system. This helps ensure that the final solution meets the stakeholders' needs while being feasible and realistic to implement [56].

This thesis aims to deliver a minimum viable product within a specified time frame. However, it is important to note that the resulting product may not be compatible with all operating systems, and it cannot be deployed on multiple blockchains due to current technological limitations.

Also for the same reason, the system may pose a challenge in providing continuous maintenance and support, which could result in unresolved bugs and technical issues.

Principle	Description
User authentication and authorization	A secure system should be in place to ensure that only eligible users can participate in voting.
Initiative creation and management	Users should be able to create and submit initiatives for voting.
Ballot creation and voting	The system should create ballots for each initiative and allow users to vote electronically while ensuring that each user can only vote once.
Result calculation and presentation	The system should calculate and present the voting results in real time, ensuring that they are accurate and transparent.
Security and privacy	The system should provide secure storage and protection of all voting data, including the anonymity of each user's vote.
Reporting and auditing	The system should generate detailed reports of the voting results and provide the ability to audit the voting process.
User management	The system should allow for the management of user accounts and voting privileges to ensure that only eligible voters can participate.

Table 4.2: Functional requirements of proposal voting system

All of these limitations and constraints must be considered when designing and developing the proposed electronic decentralized voting platform to ensure that the system is secure, reliable, and meets the needs of the stakeholders.

4.1.3 Non-functional requirements

Non-functional requirements are the qualities and characteristics of a system that are not related to specific tasks or functions. The non-functional requirements for the system that allows the creation and voting on initiatives using blockchain have been derived from the stakeholder analysis and the objectives. The stakeholder analysis helps to understand the needs and expectations of the different stakeholders such as members, leadership, technical staff, regulators, and the wider community. These expectations have been used to deter-

mine the non-functional requirements that the system must fulfill to meet the needs of the stakeholders. The objectives of the system also play a crucial role in defining the non-functional requirements. For example, the objective to ensure security and transparency in the voting process may lead to the requirement for the system to have robust security measures such as encryption and blockchain technology. The combination of stakeholder analysis and objectives has provided a comprehensive and well-rounded list of non-functional requirements that must be fulfilled by the system as depicted in Table 4.3.

Principle	**Description**
Accessibility and usability	System should be user-friendly and accessible with a simple and intuitive interface
Scalability	System should be able to handle a growing number of users and initiatives
Decentralization	System must rely on a distributed ledger managed by a network of nodes
Transparency	System must provide a clear and transparent record of all voting activities and initiatives
Privacy	System must allow users to cast anonymous votes to protect their privacy
Reliability	System must be reliable and consistent with high availability and minimal downtime
Performance	System must have fast and efficient processing times to tally votes quickly and accurately

Table 4.3: Nonfunctional requirements for a proposal voting system

4.1.4 User Requirements

User requirements are focused on the needs and expectations of end-users. These requirements describe the desired outcomes and benefits that the end-users expect to receive from using the product. User requirements are often expressed in terms of the tasks or activities that the end-users need to perform, and the features or functions they need to access.

User requirements for an electronic decentralized proposal voting platform based on blockchain are the following:

- Users should be required to securely log in with a unique identification.
- Users should have a user-friendly interface that makes it easy for them to participate in voting processes.
- Users should be able to create, submit, and vote on proposals.
- Users should be provided with real-time updates on the status of proposals and the voting process.
- Users should be allowed to view the details of each proposal and the results of each vote.
- Users should be able to flag inappropriate content and report it to moderators.

4.1.5 Technical Requirements

Technical requirements are a set of specific, technical capabilities, characteristics, and features that a product, system, or service must possess in order to fulfill its intended purpose or meet a specific set of customer or end-user needs.

Technical requirements may include hardware and software specifications, performance and capacity requirements, data management and security requirements, and other factors that are critical to the successful operation and implementation of a product or system.

Technical requirements are used to guide the development, design, and testing of products and systems, and are typically documented and agreed upon between stakeholders before development begins.

The following technical requirements are proposed in this thesis:

- Compatibility with popular web browsers and operating systems.
- Integration with blockchain technology to ensure immutability and transparency of voting data using the Ethereum network.
- Secure user authentication and authorization using the MetaMask browser extension.
- High availability and robustness to handle heavy usage and potential malicious attacks deploying the network on the Internet.

- Use of open-source software to ensure accessibility and flexibility for future development.
- Automated testing and continuous integration/delivery processes to ensure quality and speed of delivery
- Adherence to industry standards and best practices for software development and security.
- Scalability to accommodate growth in user numbers and data volume.

4.2 Architecture Design

This section provides a comprehensive overview of the system's components, their interrelationships, and the underlying structure of the software application. This section outlines the overall architecture, including high-level design decisions, technologies and tools to be used, and the relationships between components. The architecture design serves as a blueprint for the development, implementation, and deployment of the system, ensuring that all the functional and non-functional requirements are met. This section is essential for the project's success, as it provides a clear understanding of the system's components, their interactions, and how they work together to deliver the desired results.

The visual representation provided in Fig. 4.1 clarifies that each individual who participates in the voting activities of the system operates as a node on the foundational blockchain. This is facilitated through the usage of a web application developed using React, Bootstrap, and Web3.js, which may be hosted on a centralized or decentralized server. During the course of their interaction with the web application, the voter will be prompted by their digital wallet. This digital wallet acts as the authorized signer of the relevant transaction and possesses pertinent information regarding the authentication of the voter, as well as their associated credentials, such as their respective address. As a result of this transaction, it is then recorded on the blockchain in perpetuity.

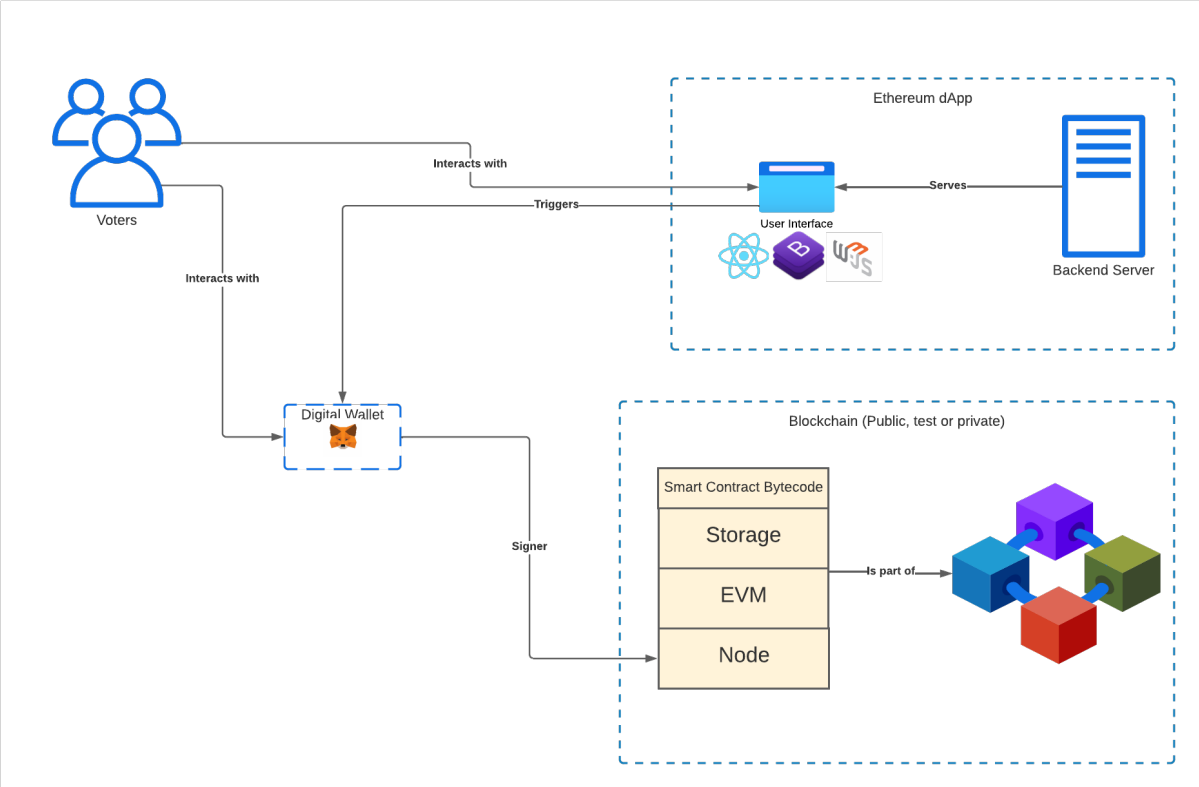


Figure 4.1: Descentralized Voting System Architecture

4.2.1 UML Diagrams for modeling the proposed system

Following the work present in [57], they noticed similarities between object-oriented programming languages and smart contracts (SCs) mainly because they have internal variables, and public and private functions able to access these variables. However, Solidity has no true classes, only smart contracts, and two defined data collection mappings and arrays.

To address these limitations and to represent the structure and relationships of SCs, the article recommends using a subset of UML diagrams, particularly the Class diagram and Sequence diagram, with the addition of specific stereotypes to capture Solidity-specific concepts. The Class diagram can effectively model the multiple inheritance, messaging, and data structure relationships among SCs and structs. Meanwhile, the Sequence diagram is useful in modeling messaging interactions among SCs and external actors, particularly transactions and calls of public functions.

Following this reasoning Table 4.4 presents the stereotypes used to define the Class

diagram (see Fig. 4.2).

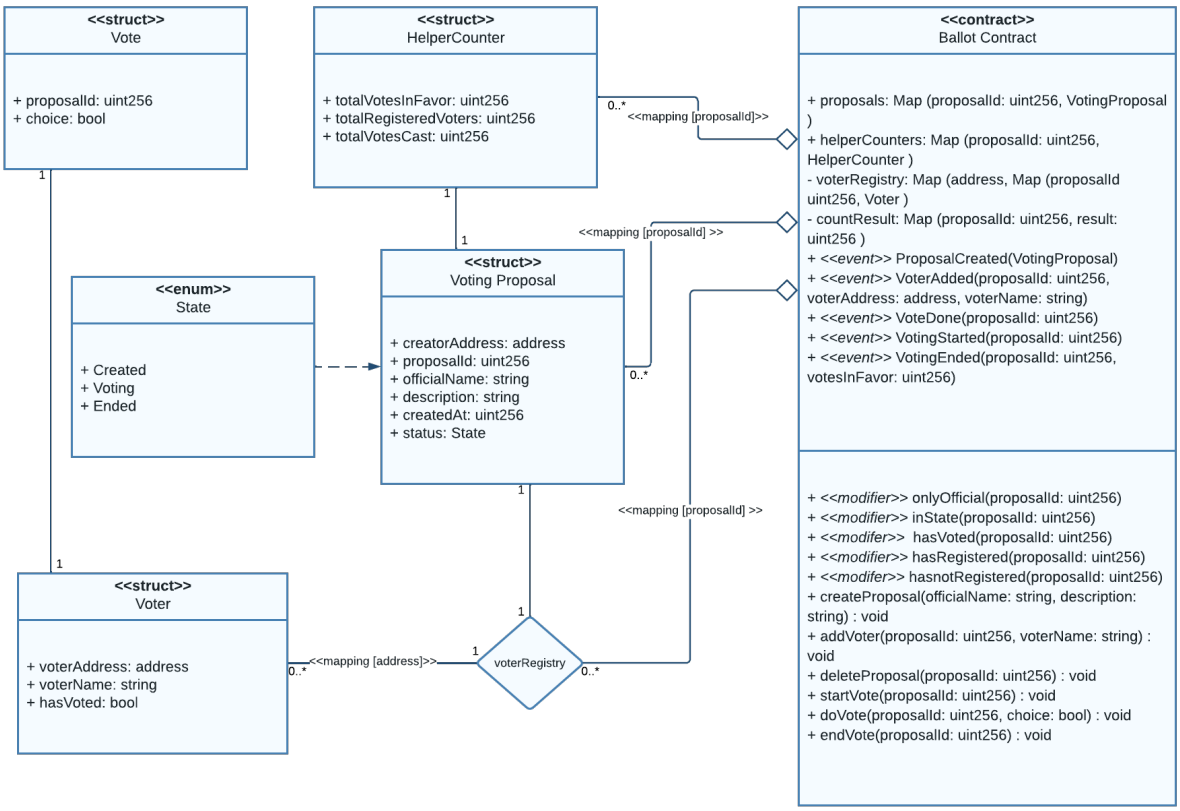


Figure 4.2: UML Class Diagram for SC for an Electronic Decentralized Proposal Voting System

A user story diagram is a visual representation of the user stories and their relationships within a system. The user story diagram in Fig 4.3 in the development of a proposal voting system can help validate the requirements of the system, facilitate collaboration between stakeholders, prioritize user stories based on their importance, and provide a basis for testing the system. Overall, a user story diagram is a useful tool in the development of any system, as it helps to ensure that all user stories are complete, consistent, and understood by all stakeholders.

4.3 Introduction to tools for development

In this section, we will describe the tools used to develop a Blockchain-based Electronic Voting System. For blockchain development, frontend development, testing, and deploying.

Blockchain development tools used in this project:

Stereotype	Position	Description
«contract»	Class symbol - upper compartment	Denotes a SC. May also be «abstract contract»
«interface»	*ditto*	A kind of contract holding only function declarations
«enum»	*ditto*	A list of possible values, assigned to some variable. The values are listed in the middle compartment. There is no bottom compartment (holding operations).
«struct»	*ditto*	A record, able to hold heterogeneous data. The fields are listed in the middle compartment. There is no bottom compartment.
«event»	Class symbol, middle compartment	An event that can be raised by an SC's function, signaling something relevant to external observers.
«modifiers»	Class symbol, bottom compartment	A particular kind of guard function, called before another function
«mapping» [*data type*]	*ditto*	The multiple variable, or the 1:n relationship, is implemented using a generic mapping.

Table 4.4: Addition stereotypes for UML Diagrams

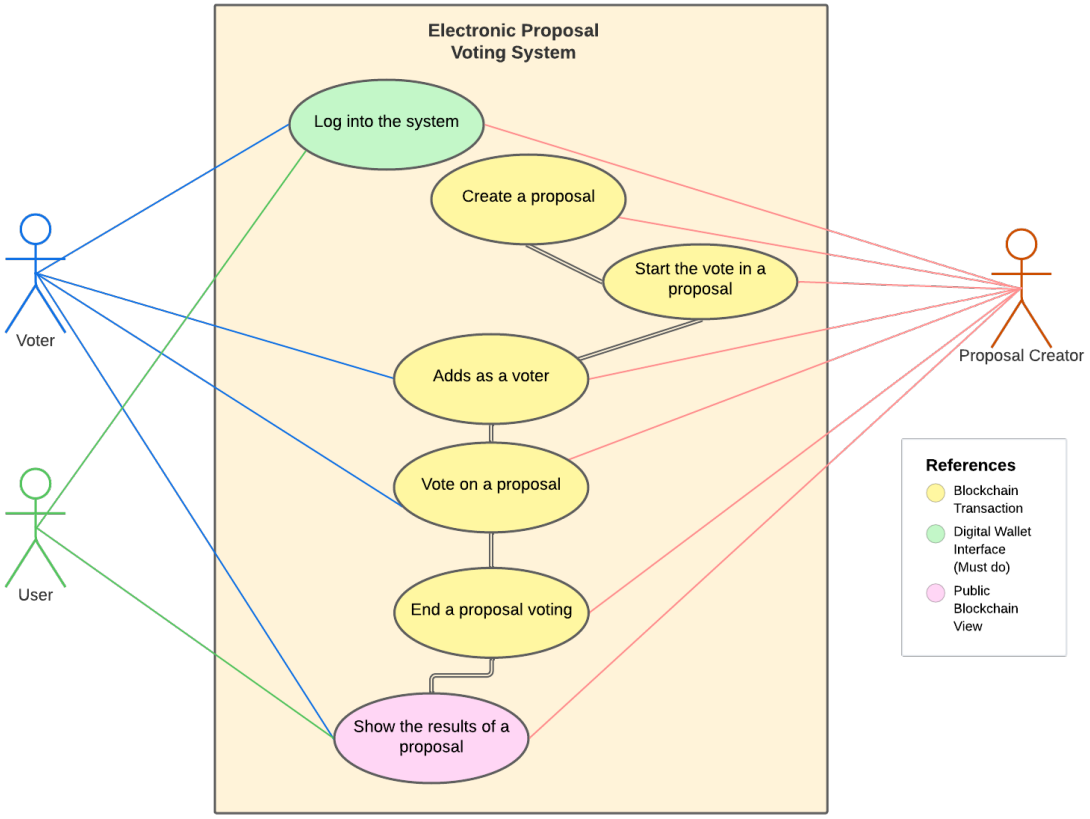


Figure 4.3: UML User Story Diagram for an Electronic Decentralized Proposal Voting System

4.3.1 Git

Git is a free and open-source distributed version control system that is widely used in software development. It allows developers to collaborate on a project by keeping track of changes made to code over time. With Git, developers can work on separate copies of the code, make changes, and merge those changes back into a shared repository. This makes it easier to manage multiple versions of the codebase, track bugs, and roll back changes if necessary. Additionally, Git provides features like branching and tagging, which allow developers to work on new features and experiment with different versions of the code without affecting the main codebase [58].

4.3.2 Truffle Framework

Truffle is an open-source tool for blockchain development that operates on the Ethereum Virtual Machine (EVM). It offers a set of tools aimed at simplifying the development

process for blockchain applications. The platform features built-in smart contract compilation, linking, deployment, and binary management capabilities. Additionally, it offers advanced debugging tools like breakpoints, variable analysis, and step-by-step execution to streamline the development process [59].

4.3.3 Ganache CLI

Ganache is a tool for quickly developing and testing Ethereum-based applications. It provides a safe and deterministic environment for the creation, deployment, and testing of these applications [60].

This thesis uses the command line interface of ganache to start a local development blockchain with a determined number of users.

4.3.4 MetaMask

MetaMask is a browser extension and a mobile app designed to provide secure management of Ethereum private keys. It serves as a wallet for Ether and other tokens and enables interaction with decentralized applications (dApps). MetaMask prioritizes user privacy by not storing personal information, including email addresses, passwords, or private keys. The user has complete control over their crypto-identity [61].

This thesis uses the MetaMask browser extension to manage transactions and authentication through the blockchain

4.3.5 Node

Node.js is a JavaScript runtime environment. It is an open-source, cross-platform platform for running web applications beyond the client's browser. The use of Node.js by developers is mainly for developing server-side web applications. It excels in data-intensive applications because of its asynchronous, event-driven architecture. As a result, Node.js has become ubiquitous in web development.

Node.js is crucial in building modern web applications, particularly front-end development. Many popular front-end tools and libraries that this thesis uses, such as Webpack and Truffle, rely on Node.js to operate.

4.3.6 React

React is a JavaScript framework created by engineers at Facebook to simplify the process of developing complex user interfaces. The goal of React was to create a scalable and maintainable framework, able to handle the demands of a large platform like Facebook. React introduces new paradigms and best practices in web development and comes with a rich set of features that make it accessible to developers of all skill levels. The framework's virtual DOM, JSX, and Flux concepts are some of the unique features that set it apart from other frameworks and enable developers to create sophisticated user interfaces. It is used for creating single page applications [62].

JSX is a way of writing HTML inside Javascript, React addresses this relationship by using components, which are self-contained units that incorporate both markup and logic. While it is not required, many developers find using JSX helpful for visually organizing their UI within the JavaScript code. The use of JSX also enables React to provide more detailed error and warning messages [63].

The UI of this thesis will be created using React as it is a robust and modern framework it will be used for rendering logic and UI interactions and will help me keep the code organized and maintainable through the use of components.

4.3.7 Typescript

TypeScript is an open-source programming language developed by Microsoft. It is a strict syntactical superset of JavaScript that adds optional type annotations and provides improved type checking, which helps to catch bugs before runtime and improve maintainability. TypeScript is also designed to be backward compatible with JavaScript and can be compiled to produce clean and efficient JavaScript code [64].

This thesis uses Typescript for its strong typing system, which helps to prevent type errors in your code and increases overall code quality. Additionally, TypeScript has features such as Interfaces, Classes, and Generics, which allow you to write modular, reusable, and maintainable code. Furthermore, TypeScript is a statically typed language that is a superset of JavaScript, which means it can transpile to clean and simple JavaScript that can run in any JavaScript environment. With its features and compatibility, TypeScript

makes it easier to develop and scale large and complex applications. This will be the programming language used for React.

4.3.8 Webpack

Webpack is a popular open-source module bundler for modern web applications. It allows developers to organize and bundle different types of files, such as JavaScript, CSS, and images, into a single package that can be loaded efficiently by a web browser.

The purpose of using Webpack is to optimize the performance and efficiency of web applications. It can help to minimize the number of requests made to the server, reduce the size of the files that need to be loaded, and ensure that the code is executed in the correct order. Webpack also supports features such as code splitting and tree shaking, which further improve the performance of web applications.

In addition, Webpack provides a range of plugins and loaders that can be used to customize the bundling process and add additional functionality to a web application. This can help to simplify the development process and improve the quality of the final product.

In the case of the thesis, the contracts ABI will also be served by Webpack, facilitating the main communication between the frontend and the blockchain.

4.3.9 Bootstrap

Bootstrap is a popular front-end web development framework that provides pre-built HTML, CSS, and JavaScript components to help developers quickly create responsive and mobile-friendly websites.

Bootstrap and Bootstrap React are web development libraries that provide pre-built components and layouts for creating responsive and modern websites and web applications. Using these libraries can save developers time and effort, as well as provide a highly customizable foundation for building visually appealing designs. Both libraries have active communities that contribute new features and knowledge, making it easier for developers to stay up-to-date with best practices.

4.3.10 Docker

Docker is an open-source platform designed to create, deploy, and manage containerized applications. It enables developers to package an application with all its dependencies into a container, making it easy to move between environments and deploy consistently.

This thesis uses Docker to facilitate the creation and deployment of the electronic voting platform, simplify the development environment, and improve reproducibility. Utilizing Docker, we can standardize the development and deployment process without worrying about system configuration issues.

4.3.11 Web3.js

Web3.js is a JavaScript library that provides developers with a way to interact with the Ethereum blockchain. It enables them to write applications that can communicate with the blockchain, allowing them to access data, send transactions, and interact with smart contracts.

In this thesis, we will be using this library so users can interact with the smart contract through an API. This API helps us communicate the smart contract and Metamask browser extension.

4.4 User Interface Design

In terms of the User Interface, the objective was to create a straightforward interface that would offer accessibility and easy navigation for all users. The design drew inspiration from Single Page Applications (SPAs) such as Netflix and other streaming platforms, which demonstrate a seamless browsing experience without the need for server communication to provide HTML content. Rather, the HTML content is generated reactively in response to user input.

Then, we used an online tool Figma to design a SPA with this principle in mind as seen in Fig. (4.4).

The system's underlying design principle is characterized by its straightforwardness. Specifically, the interface is divided into two distinct regions, each serving a different purpose. On the left side of the interface, information pertaining to the currently logged-in

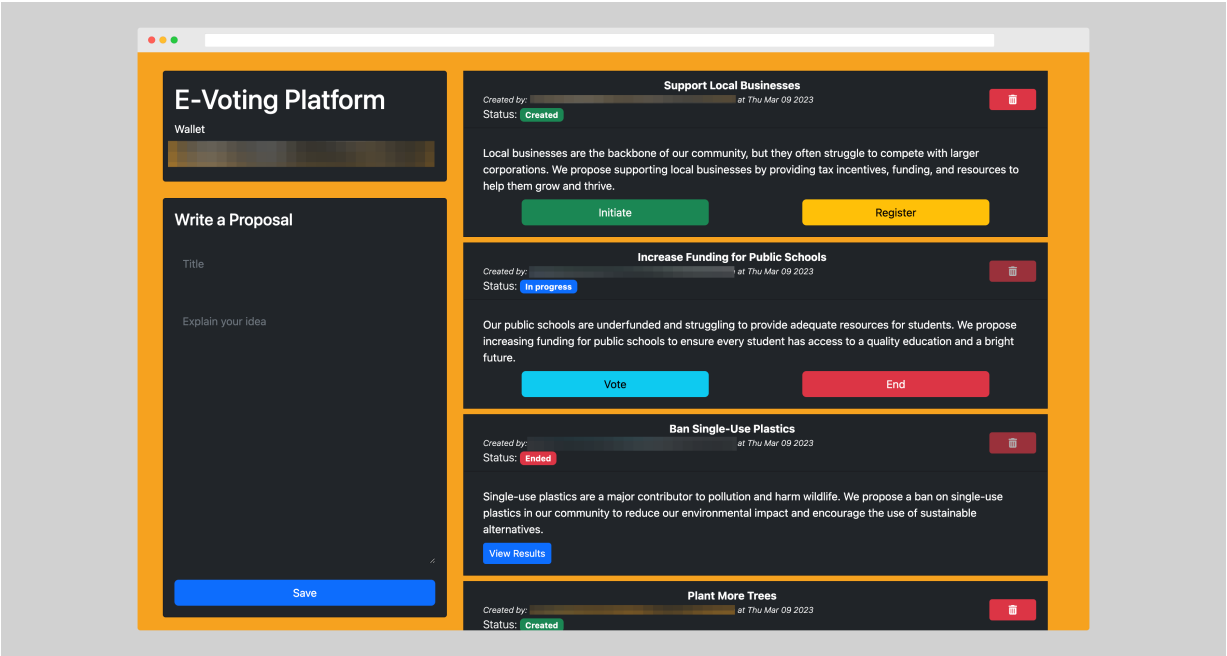


Figure 4.4: User Design Mockup: Proposal List

account is provided, accompanied by a form below that enables users to create new proposals. Conversely, the right side of the interface displays a list of all created proposals and is equipped with a set of buttons that facilitate specific actions, such as Starting, Registering, Voting, and Ending the proposals. Upon clicking these buttons, a modal is displayed to confirm the corresponding action on the blockchain. Additionally, when a proposal reaches the Ended state, a new button is generated to showcase the results on a modal, as illustrated in Fig. (4.5).

As this thesis is still a work in progress and lacks access to actual users for usability testing, alternative methods can be explored, including heuristic evaluation of the system.

4.4.1 Heuristic Evaluation

A heuristic evaluation can be conducted by employing a set of established usability principles or heuristics to assess the interface, a heuristic evaluation can help identify potential usability issues and improve the overall user experience. This evaluation aims to assess the electronic voting system against commonly recognized usability heuristics to identify areas for improvement and provide recommendations [65].

The following assessment was made to the system in order to improve usability. The

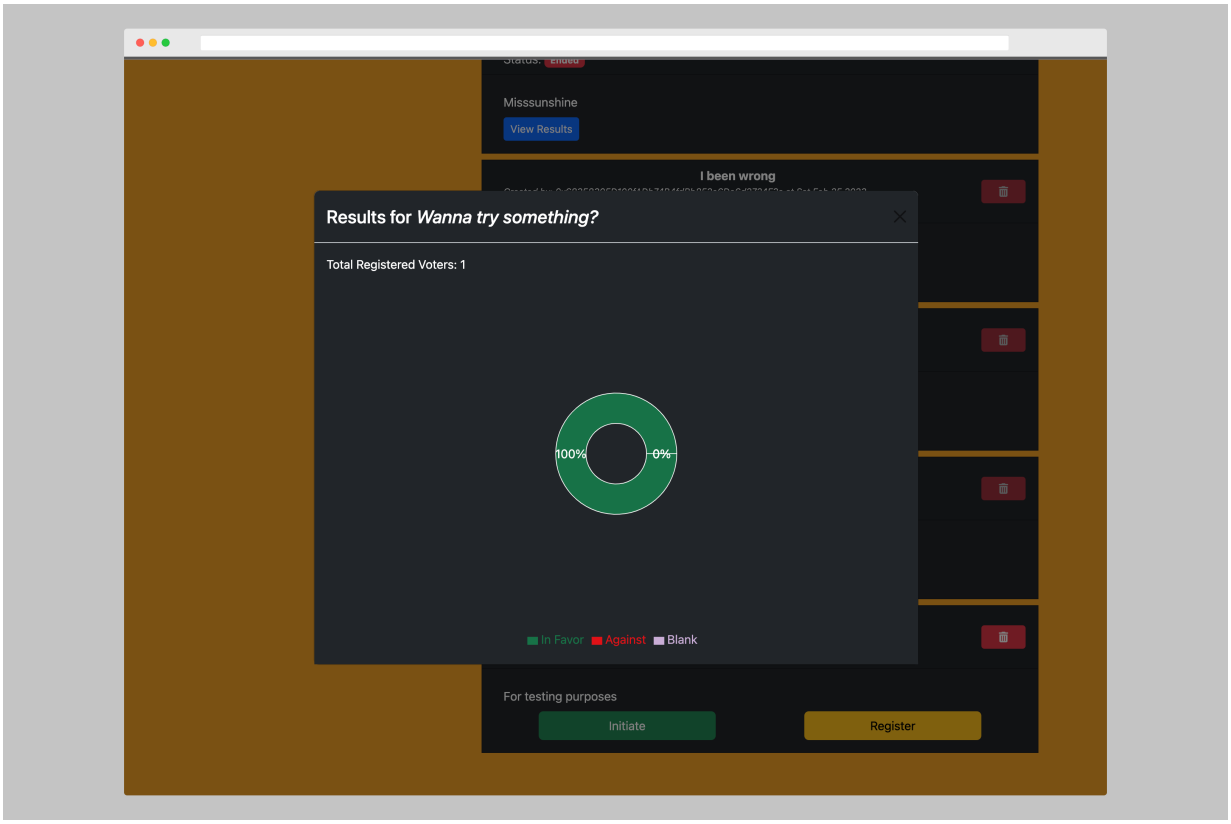


Figure 4.5: User Design Mockup: Results View

next iterations of the system should accomplish passing these tests. The test is based on a literature review of electronic voting systems [66].

The heuristic evaluation of the electronic voting system highlighted several areas for improvement to enhance its usability and user experience. Recommendations include improving feedback mechanisms, aligning the system's terminology with users' mental models, providing more user control and flexibility, ensuring consistency in design and interaction patterns, preventing and guiding users through errors, and offering accessible help resources. By addressing these recommendations, the electronic voting system can improve its usability, resulting in a more efficient and satisfying experience for voters.

Test	Result
Ensure that the system provides clear and timely feedback to users about the current status of their actions, such as confirming vote submission or indicating errors	✓
Display appropriate progress indicators or notifications to keep users informed about the progress of the voting process	✓
Use language, terminology, and concepts familiar to voters to minimize confusion and ensure a clear understanding of the voting process	✓
Provide clear instructions and guidance to help users navigate the system and make informed choices	✗
Allow users to easily correct errors or change their votes before final submission	✓
Provide clear options for users to exit or cancel the voting process without unintended consequences	✗
Ensure consistent design and interaction patterns throughout the system to reduce cognitive load and make the system more intuitive	✓
Adhere to established design conventions and standards to ensure familiarity and ease of use for users	✓
Implement mechanisms to prevent common voting errors, such as preventing users from submitting multiple votes or invalid selections	✗
Provide clear error messages and guidance to help users rectify any mistakes or issues	✗
Minimize the need for users to remember complex instructions or previous steps in the voting process	✓
Clearly present options and choices to users at each step to reduce reliance on memory	✓
Design the system to accommodate a range of user abilities and preferences, allowing both novice and experienced users to interact with the system effectively	✓
Provide shortcuts or advanced features for experienced users to expedite their interaction with the system	✗
Strive for a visually pleasing and clutter-free interface, focusing on essential information and minimizing distractions	✓
Use appropriate visual hierarchy, typography, and colors to guide users' attention and enhance the overall aesthetics	✓
Clearly communicate error messages in plain language, avoiding technical jargon	✗
Provide actionable guidance to help users understand and resolve errors	✓

Table 4.5: Assessing system usability through Heuristic Evaluation

Chapter 5

Results and Discussion

This chapter introduces several sections on the development, testing, and deployment of the system. As the results of this thesis are a proof-of-concept implementation of the proposed system, it is important to detail the development process, including the technologies used and the design decisions made. Additionally, testing played a crucial role in ensuring the quality and reliability of the system. In this chapter, we discuss the various testing approaches used, including unit testing, integration testing, and system testing. Finally, we address the deployment of the system and the challenges and considerations involved in deploying a blockchain-based application. The sections provide a comprehensive overview of the development, testing, and deployment phases of the project and serve as a guide for future researchers or developers who may be interested in building on this work.

1. Development Process: General steps and processes followed throughout the agile development cycle. It also addresses any challenges or obstacles faced during the development process and the methods used to overcome them.

2. Implementation: Description of the steps taken to implement the system.

3. Deployment: Explanation of the process of deploying the system, including any challenges faced and the methods used to overcome them.

4. Testing: Explanation on tests implemented to accomplish a Test Driven Development(TDD).

5.1 Development Process

The development process of this thesis followed the agile approach to development focusing on a minimum viable product that let users create and vote on initiatives. The agile approach consists in having these ceremonies such as sprint planning, daily meeting, sprint review, and retrospectives. As a one-person team, one person had to take on the responsibilities typically assigned to multiple roles, such as product owner, scrum master, and developer. So given that we defined what the backlog was following the user story principle for the tasks that needed to be performed (see Fig. 5.1).

Once the backlog was defined, the next steps were to divide the work into sprints that we set up to be short, time-boxed periods of two weeks to finish the tasks. An example of how a sprint looked like (see Fig. 5.2)

The whole project took about 12 sprints to complete following the attached Grantt diagram (see Fig. 5.3). As a one-person team, it was not required to have a daily sync but sprint reviews and retrospectives were most important. The tutor of this thesis served the role of a product owner on a weekly meeting to report development progress as well as provide meaningful insights into the development of this project. In the sprint review meeting, we would refine the backlog as well as add more user stories as we started to get more knowledge about the technologies used and the application design.

Initially, the process of estimating the work presented challenges due to the lack of prior experience. However, after obtaining a better understanding of the project's scope and requirements, the sprint reviews proved to be instrumental in accurately estimating user stories on the backlog. Through ongoing grooming, we were able to incorporate additional acceptance criteria and consider edge cases that were not initially identified. The same issue of no prior experience led to the accumulation of technical debt, which was necessary as we had a functional product but identified areas for improvement, such as the use of Docker for development. This issue became apparent when we had to switch to a different environment, which proved to be a time-consuming task that hindered our progress toward the minimum viable product. Nonetheless, this obstacle helped us to recognize the importance of identifying and prioritizing tasks that would streamline the development process and promptly add them to the backlog for future iterations.

The main pressurizing challenge was the lack of input from other stakeholders as the one-person team had to take the role of the developer, the elected official, and the community. We struggled to develop a platform that would satisfy all the requirements these roles would have in a real-life scenario. Nonetheless, the constant iterative nature of agile helped us to realize areas of improvement for users to make the application easier to use. For example, the process of setting up a node is complex but rewarding. As it is a key pillar of blockchain technology improvements would have to be done for the next iterations of the product.

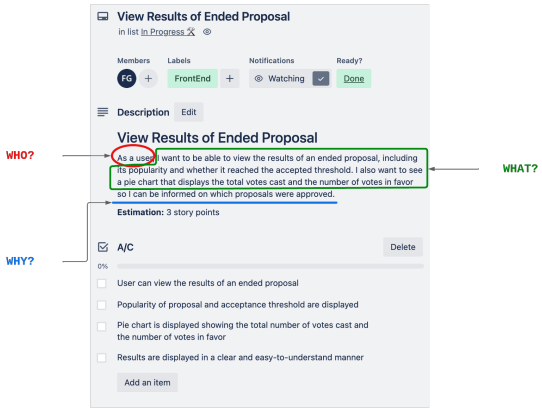


Figure 5.1: User story about results view

5.2 Implementation

5.2.1 Setting up the Development Environment

This section will provide a detailed overview of how we set up the development environment for the blockchain voting system, including the installation and configuration of Ganache and Truffle.

To facilitate development, we created a monorepo that holds the code for both the smart contracts and the front end of the application. To start up the development environment, we provided two approaches: Local Development and Docker Development.

Local Development involves setting up the development environment on the local machine. This approach requires installing and configuring the necessary software components, including Ganache and Truffle, and running the application locally.

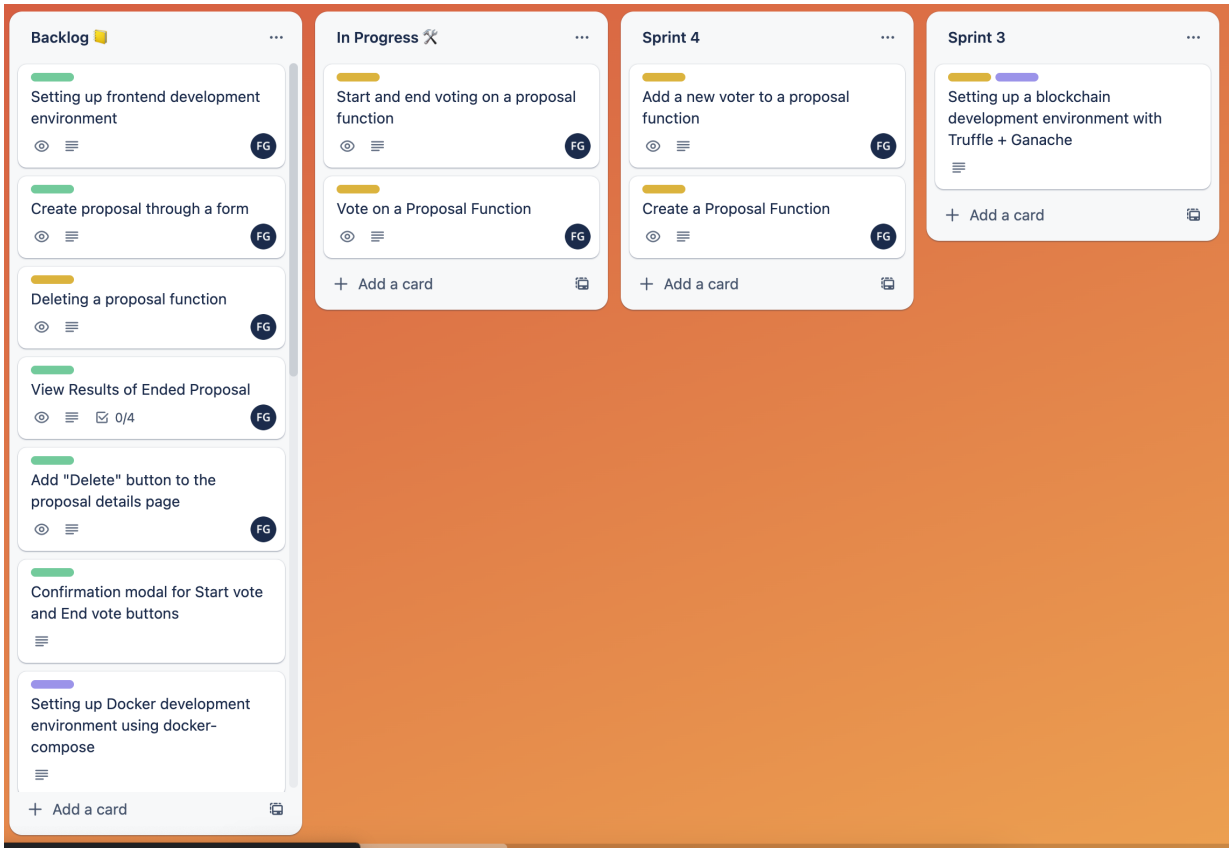


Figure 5.2: Sprint 4 - Developing smart contract

PROJECT: Electronic Voting Platform

Graduation Project

Franz Guzmán

Project start date: 11/11/21

FrontEnd | Backend | Design | Infrastructure | Tech Debt

Milestone description	Category	Progress	Start
Project Setup	Milestone	100%	11/11/21
Smart Contract Development	Milestone	100%	23/12/21
Create a Proposal Function	Backend	100%	23/12/21
Add a new voter to a proposal function	Backend	100%	23/12/21
Adding code linter rules and formatting rules	Tech Debt	100%	23/12/21
Vote on a Proposal Function	Backend	100%	6/1/22
Start and end voting on a proposal function	Backend	100%	6/1/22
FrontEnd Development	Milestone	100%	6/1/22
Create proposal through a form	FrontEnd	100%	6/1/22
Setting up frontend development environment	FrontEnd	100%	20/1/22
Status Badges for Proposal	FrontEnd	100%	20/1/22
Create modal for in-browser voting	FrontEnd	100%	3/2/22
Docker and Improvements	Milestone	100%	3/2/22
Setting up Docker development environment using docker-compose	Infrastructure	100%	3/2/22
Add Events for Solidity Functions	Backend	100%	3/2/22
Migrating to TypeScript	Tech Debt	100%	3/2/22
Deleting a proposal function	Backend	100%	17/2/22
Connect Frontend to Smart Contract through MetaMask	FrontEnd	100%	17/2/22
Improve Styling and UX	FrontEnd	100%	17/2/22

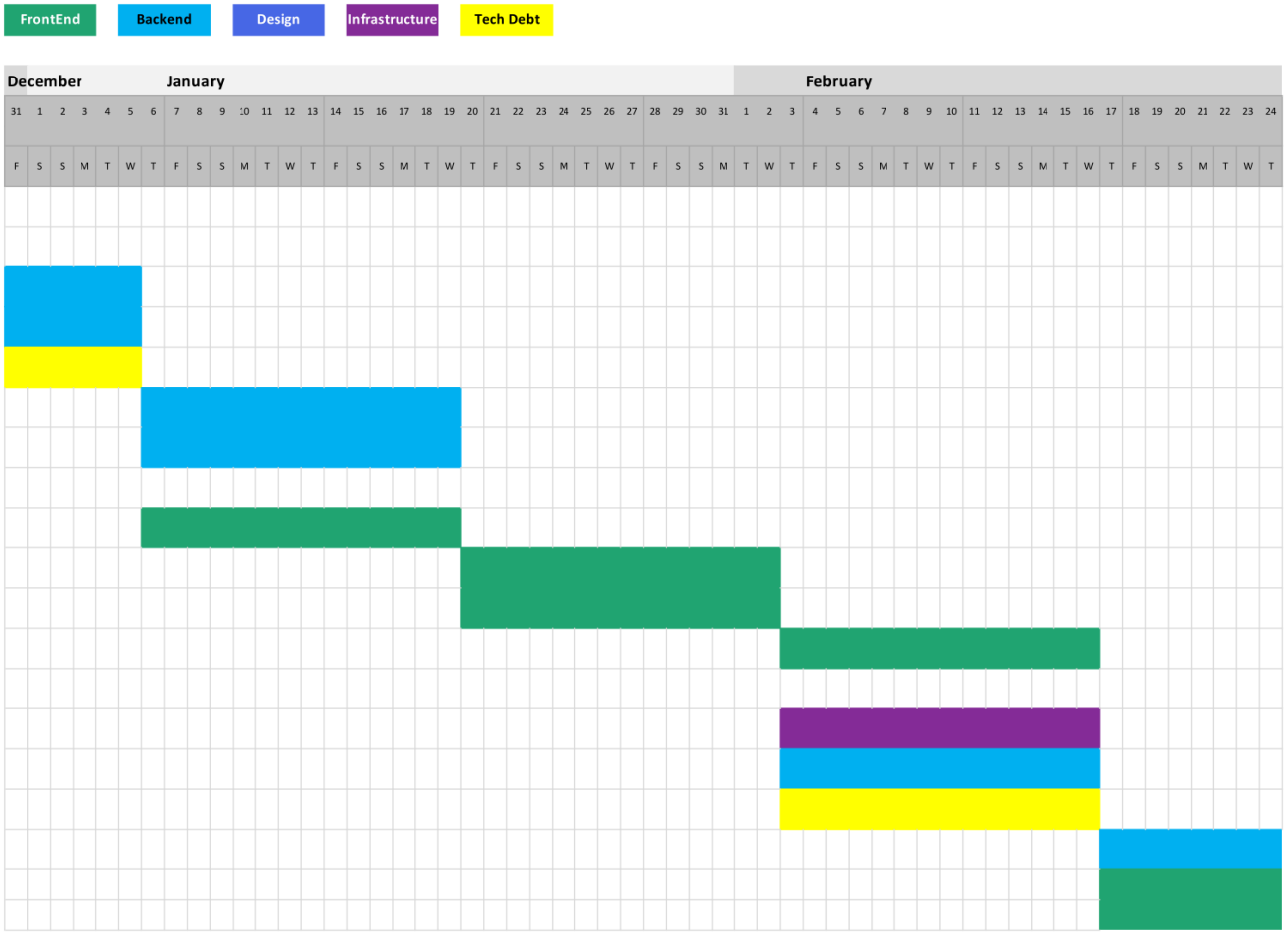


Figure 5.3: Gantt Diagram of the Development Process

On the other hand, Docker Development utilizes Docker containers to create a self-contained development environment. This approach allows for easier setup and configuration of the development environment and ensures consistency across different machines and operating systems.

In the following subsections, we will provide detailed instructions on how to set up the development environment using both approaches.

For **Local Development**:

1. First to utilize Ganache as a local Ethereum network, it is necessary to install the software. This can be accomplished by downloading it from the official website[1] to use the Graphical User Interface (GUI) version, or by installing the package through the use of the following command:

```
1          npm install -g ganache
```

2. Next it is necessary to install truffle in the local environment to be able to compile, migrate and test the contracts in the local Ethereum network.

```
1          npm install -g truffle
```

3. In order to develop the frontend part of the project, a subdirectory was created. To ensure that the dependencies required for this subdirectory are installed.

```
1          cd frontend && npm install
```

   This command does two things:

   - It changes the current directory to the "frontend" subfolder, which is where the frontend development is being done.
   - It installs the necessary dependencies listed in the `package.json` file, which is a file that contains information about the project and its dependencies.

4. We now have all the requirements installed the next step is to run the ganache network with this command:

[1] https://trufflesuite.com/ganache/

```
ganache -h 0.0.0.0 -p 8545
```

It launches an instance of the Ganache blockchain emulator with the following parameters:

- -h 0.0.0.0: This sets the host IP address that Ganache listens on to 0.0.0.0. This means that Ganache is accessible from any device on the network.
- -p 8545: This sets the port number that Ganache listens on to 8545. This is the default port number used by Ethereum clients to communicate with the blockchain.

5. We will have to set up the truffle configuration file `truffle-config.js` with the following configuration:

```
module.exports = {
      networks: {
        local_development: {
          host: "127.0.0.1",
          port: 8545,
          network_id: "1",
        },
        development: {
          host: "ganache-cli",
          port: 8545,
          network_id: "1",
        },
        testing: {
          host: "127.0.0.1",
          port: 8546,
          network_id: "2",
        },
        production: {
          host: "0.0.0.0",
          port: 8548,
          network_id: "100",
        },
      },

    compilers: {
      solc: {
        version: "0.8.9",
```

```
28          },
29        },
30      };
```

The file exports a JavaScript object with two main sections, networks which section defines the various networks that the smart contract can be deployed to, along with their configuration options. `local_development`, `development`, `testing` and `production`. compilers section defines the compiler version used to compile the smart contracts.

6. Once truffle is configured we are able to run the migrations to the network with this command:

```
1          truffle migrate --network local_development
```

This command first compiles our contracts and then performs the migrations to the network specified after the `network` flag. It will generate the contract's ABI which will be used to interact with the network.

7. Finally, we can boot up the frontend which can be done with the following command inside the `frontend/` subfolder:

```
1          npm run dev
```

This command tells Webpack to compile all our frontend and assets and serve them on the development server on port 8080.

This is the process to start developing smart contracts and the frontend of the application.

For **Docker Development**, it gets simpler all this previous configuration gets abstracted from the developer in a single `docker-compose.yml` file. This file simplifies the process of defining a complex multi-container application by allowing me to declare all the necessary components and their dependencies in a single file. This file looks as follows:

```
1 version: '3'
2
3 services:
4   ganache-cli:
5     image: trufflesuite/ganache-cli
6     container_name: ganache_cli
7     command: ganache-cli -h 0.0.0.0 -p 8545 -i 1
```

```
8       ports:
9         - "8545:8545"
10    truffle:
11      container_name: truffle
12      build:
13        context: .
14        dockerfile: truffle.Dockerfile
15      depends_on:
16        - ganache-cli
17      tty: true
18      ports:
19        - "8080:8080"
20      command: npm run dev
```

This configuration defines two services: ganache-cli and truffle.

The first service, ganache-cli, uses the trufflesuite/ganache-cli image from Docker Hub to run a local Ethereum blockchain. It also exposes port 8545 to the host machine and starts the ganache-cli command with specified options.

The second service, truffle, builds an image from a Dockerfile located in the current directory. This image was described with the same steps performed previously. This service depends on the ganache-cli service to be running first. The service opens port 8080 for external access and starts a command that runs the starts development server for the web application.

Then a simple command builds the containers from the images and runs the services,

```
1       docker-compose up
```

5.2.2 Smart Contract Development

The smart contract manages the rules and procedures for conducting the voting process and ensures the integrity of the results. The section provides an overview of the development process for the smart contract, written in Solidity, the programming language for writing smart contracts in Ethereum. Solidity is specifically designed for writing smart contracts and is well-suited for developing decentralized applications on the Ethereum blockchain. We also provide a detailed description of the smart contract's code structure and the functions it has.

The contract code has the following key features:

- It defines a public struct called VotingProposal to store information about each proposal, such as the creator's address, proposal name, description, date created, and status.
- It defines a private struct called Voter to store information about each registered voter, such as their address, name, and whether they have voted.
- It defines a private struct called Vote to store information about each vote, including the proposal ID and the voter's choice (in favor or not in favor).
- It defines an enum called State to represent the current status of each proposal, which can be **Created**, **Voting**, or **Ended**.
- It defines several modifiers to restrict access to specific functions based on the proposal's state or the voter's status.
- It defines several events to notify interested parties when a new proposal is created, a voter is added, a vote is cast, or a proposal is ended.
- It defines several functions which will be examined in the following paragraph.

The functions defined in the contract allow for the creation of proposals, the registration of voters, the deletion of proposals, the start and end of the voting process, and the casting of votes.

In Fig. (5.4) there is a graphical representation of the main data flow of the smart contract. This flowchart illustrates the various functions and operations involved in the execution of the contract, as well as the interactions between different entities such as the contract itself, external data sources, and the users of the system.

The main functions are the following:

```
1 function createProposal(
2     string memory _officialName,
3     string memory _description
4 ) public {
5     proposalCounter++;
6     proposals[proposalCounter] = VotingProposal(
```

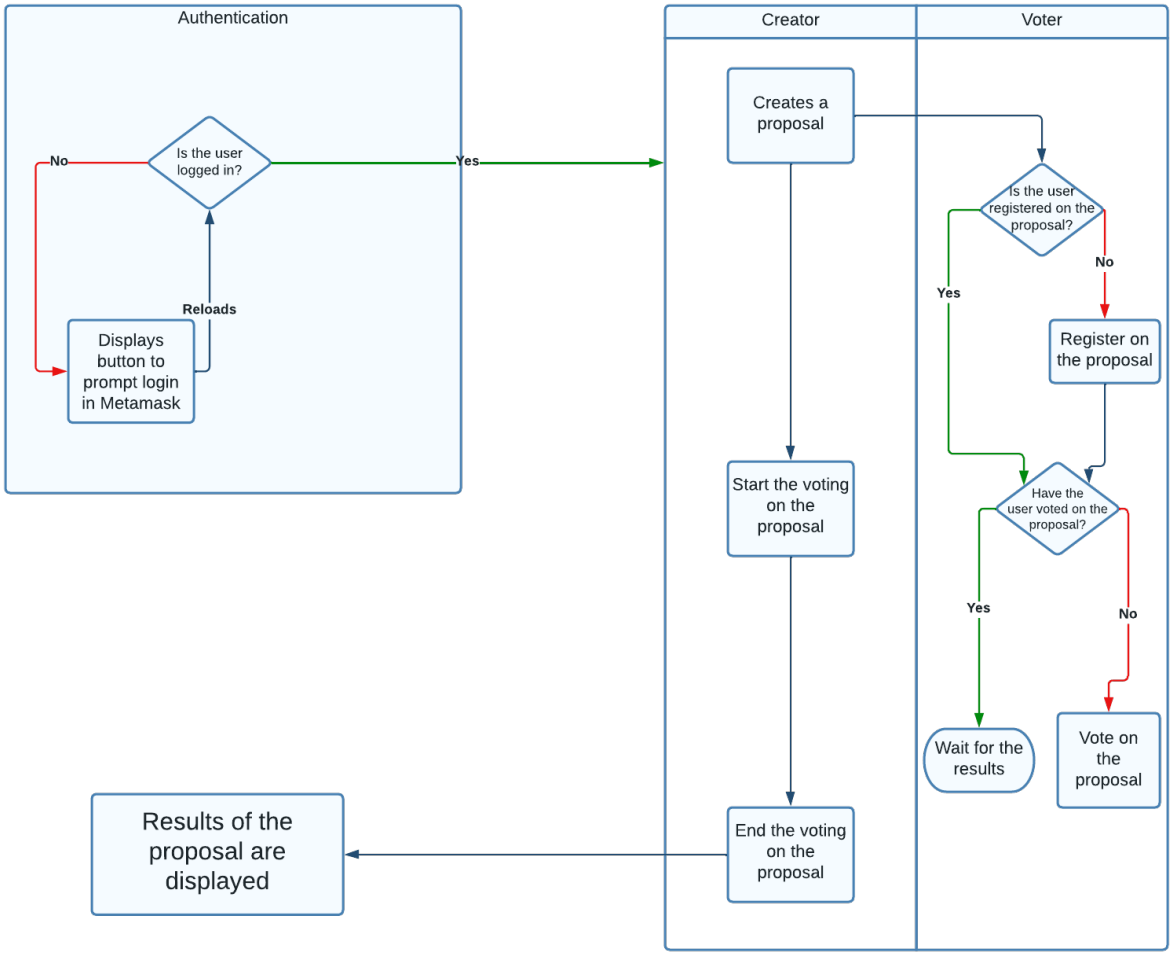


Figure 5.4: User Flow

```
7           msg.sender,
8           proposalCounter,
9           _officialName,
10          _description,
11          block.timestamp,
12          State.Created
13      );
14      helperCounters[proposalCounter] = HelperCounter(0, 0, 0);
15      emit ProposalCreated(
16          proposalCounter,
17          msg.sender,
18          _officialName,
19          _description,
20          block.timestamp,
21          State.Created
22      );
23  }
```

The code is designed to create a new proposal, it takes two arguments, the `officialName` of the proposal and the `description` of the proposal. Then stores the proposal, including the sender's information, current timestamp, and status as "Created," in the proposals mapping. It also creates a HelperCounters entry for the proposal and emits a ProposalCreated event that includes information about the proposal.

```
1  function addVoter(uint256 _proposalId, string memory _voterName)
2      public
3      inState(State.Created, _proposalId)
4      hasNotRegistered(_proposalId)
5      hasNotVoted(_proposalId)
6  {
7      helperCounters[_proposalId].totalRegisteredVoters++;
8      voterRegistry[msg.sender][_proposalId].voterAddress = msg.sender;
9      voterRegistry[msg.sender][_proposalId].voterName = _voterName;
10     voterRegistry[msg.sender][_proposalId].hasVoted = false;
11     emit VoterAdded(_proposalId, msg.sender, _voterName);
12 }
```

This function takes two arguments - `proposalId`, which is the ID of the proposal to which the voter is being added, and `voterName`, which is the name of the new voter being added.

The function also contains three modifiers - `inState`, `hasNotRegistered`, and `hasNotVoted`. These modifiers are used to ensure that the voter being added is eligible to vote on

the proposal. And also the proposal is in the correct state. If the voter satisfies all three modifiers, then the function executes and performs the following tasks. These modifiers are useful instead of if/else chains inside the function code.

1. It increments the totalRegisteredVoters counter in the helperCounters mapping associated with the given proposal ID.
2. It creates a new entry in the voterRegistry mapping, which stores the voter's address, name, and whether they have voted or not.
3. It emits a VoterAdded event, which contains information such as the proposal ID, and the voter's address.

```
1  function deleteProposal(uint256 _id)
2      public
3      inState(State.Created, _id)
4      onlyOfficial(_id)
5  {
6      delete proposals[_id];
7  }
```

This is a descriptive function it requires the ID of the proposal as an argument and removes that proposal if the two modifiers are correct, the executor of this function is the one that created the proposal and the proposal must be in state **Created**.

```
1  function startVote(uint256 _id)
2      public
3      inState(State.Created, _id)
4      onlyOfficial(_id)
5  {
6      VotingProposal memory _proposal = proposals[_id];
7      _proposal.status = State.Voting;
8      proposals[_id] = _proposal;
9      emit VotingStarted(_id);
10 }
```

This function starts the vote on the given proposal ID. It changes the status from Created to **Voting**. It also emits an event called VotingStarted, with the ID of the proposal such that the frontend can handle the state change of the blockchain in real-time.

```
function doVote(uint256 _id, bool _choice)
    public
    hasRegistered(_id)
    hasNotVoted(_id)
    inState(State.Voting, _id)
{
    voterRegistry[msg.sender][_id].hasVoted = true;
    Vote memory v;
    v.proposalId = _id;
    v.choice = _choice;
    if (_choice) {
        countResult[_id]++;
    }
    votes[_id][helperCounters[_id].totalVotesCast++] = v;
    helperCounters[_id].totalVotesCast++;
    emit VoteDone(_id);
}
```

This function requires that the proposal is in the state **Voting**, that the voter has been registered as a voter in the given proposal, and that the voter has not cast a vote before.

Then the function reads the vote and increments the private variable `countResult` if the vote was in favor of the proposal. If not then just saves the ballot in a mapping. Finally, increments the helper counter of the totalVotesCast.

After all, it emits a VoteDone event on that proposal.

```
function endVote(uint256 _id)
    public
    inState(State.Voting, _id)
    onlyOfficial(_id)
{
    VotingProposal memory _proposal = proposals[_id];
    _proposal.status = State.Ended;
    proposals[_id] = _proposal;
    helperCounters[_id].totalVotesInFavor = countResult[_id];
    emit VotingEnded(_id, countResult[_id]);
}
```

Lastly, this function performs the counting of the results and publishes them in the helper Counters mapping that is publicly accessible.

It requires that the proposal is in the Voting state. And the executor of the function is the one that created the proposal.

Finally, it emits the VotingEnded event.

5.2.3 Front-End Development

The core library that would help me achieve the desired page reactivity is **React**. To interact with the Ethereum blockchain, we used the Web3.js library, which provided a simple API for interacting with smart contracts deployed on the blockchain.

5.3 Testing

The proposed e-voting platform was evaluated through various tests and experiments. The system was developed using test-driven development (TDD), which is a way of creating software that involves writing tests before code. TDD follows a cycle of writing a test, running it, and writing code to pass it. This cycle is repeated for each feature or requirement until the software is complete.

TDD helps to make sure that the code is well-tested and meets the expected requirements before it is released. By writing tests first, developers can specify what the software should do and how it should work. This way also helps to find and fix errors early in the development process, which makes it easier and cheaper [67].

TDD also helps developers to write code that is modular, easy to maintain, and easy to test. With TDD, each test checks that previous features still work when the code changes. This way can lead to faster development, better code quality, and lower costs related to bug fixes and maintenance.

Following the agile approach, the electronic voting system is built and tested incrementally. Each non-trivial function of the contracts written so far must be provided with Unit and Acceptance Tests. Truffle framework makes available two methods for testing Ethereum smart contracts: Solidity test and JavaScript test. For the sake of brevity, here we report just a fragment of the Javascript unit test written to verify “addVoter()”, “doVote()” and “endVote()” and functions of the `BallotContract` contract:

```
1   describe("Add Voters", () => {
2     it("adds voters to the proposal", async () => {
3       // Add a voter to the proposal
```

```
4         const tx = await ballotContract.addVoter(proposalCount, "Voter 1
              ", {
5           from: accounts[1],
6         });
7         // Check if the voter has been added
8
9         truffleAssert.eventEmitted(tx, "VoterAdded", (ev) => {
10          return (
11            ev.proposalId.toNumber() === proposalCount.toNumber() &&
12            ev.voterAddress === accounts[1] &&
13            ev.voterName === "Voter 1"
14          );
15        });
16      });
17
18      it("should fail if the proposal is not in the created state",
            async () => {
19        // Start the voting on the proposal
20        await ballotContract.startVote(proposalCount, { from: accounts
              [0] });
21
22        // Try to add a voter to the proposal
23        try {
24          await ballotContract.addVoter(proposalCount, "Voter 1", {
25            from: accounts[1],
26          });
27        } catch (error) {
28          assert.strictEqual(
29            error.message.includes(
30              "Error: The proposal is not in the state required to
                    perform this action"
31            ),
32            true
33          );
34        }
35      });
36      it("should fail if the voter has already been registered", async
            () => {
37        // Add a voter to the proposal
38        await ballotContract.addVoter(proposalCount, "John Doe", {
39          from: accounts[2],
40        });
41        // Try to add the same voter again
42        try {
```

```
43          await ballotContract.addVoter(proposalCount, "John Doe", {
44            from: accounts[2],
45          });
46        } catch (error) {
47          assert.strictEqual(
48            error.message.includes(
49              "Error: You are already registered as a voter for this
                    proposal"
50            ),
51            true
52          );
53        }
54      });
```

The provided test case aims to check the functionality of the "Ballot Contract" by simulating the addition of voters to a proposal. The test case includes three scenarios; the first one checks whether a voter can be added successfully to the proposal, the second test case checks whether the system prevents adding voters when the proposal is not in the "Created" state, and the third test case ensures that the system prevents adding the same voter twice to the proposal.

```
1   describe("doVote function", () => {
2     let ballotContract;
3     let proposalCount;
4     let voterAddress;
5
6     before(async () => {
7       ballotContract = await BallotContract.deployed();
8       voterAddress = accounts[1];
9       // Add a proposal to the contract
10      await ballotContract.createProposal(
11        "Proposal Title",
12        "Proposal Description",
13        { from: accounts[0] }
14      );
15      proposalCount = await ballotContract.proposalCounter();
16      await ballotContract.addVoter(proposalCount, "Voter 1", {
17        from: accounts[1],
18      });
19      await ballotContract.startVote(proposalCount, { from: accounts
            [0] });
20    });
21    beforeEach(async () => {
```

```
22        ballotContract = await BallotContract.deployed();
23        voterAddress = accounts[1];
24      });
25      it("should allow a registered voter to cast a vote", async () => {
26        // Cast a vote for the proposal
27        const choice = true;
28        await ballotContract.doVote(proposalCount, choice, {
29          from: accounts[1],
30        });
31
32        // Check that the vote has been recorded
33        const counters = await ballotContract.helperCounters(
              proposalCount);
34        assert.equal(counters.totalVotesCast, 1, "Total votes cast is
              incorrect");
35      });
36
37      it("should not allow a voter to cast multiple votes for the same
           proposal", async () => {
38        // Try to cast another vote for the same proposal
39        const choice = false;
40        try {
41          await ballotContract.doVote(proposalCount, choice, {
42            from: accounts[1],
43          });
44          assert.fail("Expected doVote to throw an error");
45        } catch (error) {
46          assert.strictEqual(
47            error.message.includes(
48              "Error: You have already cast your vote on this proposal
                    or you are not registered as a voter"
49            ),
50            true
51          );
52        }
53        const counters = await ballotContract.helperCounters(
              proposalCount);
54        assert.equal(counters.totalVotesCast, 1, "Total votes cast is
              incorrect");
55      });
56    });
```

The “doVote” function’s test suite allows registered voters to vote for a particular proposal. The test suite contains two test cases.

The first test case checks if the “doVote” function allows a registered voter to cast a vote for a proposal. It casts a vote for a proposal and then checks if the vote has been recorded by verifying the “totalVotesCast” counter.

The second test case checks if the “doVote” function prevents a registered voter from casting multiple votes for the same proposal. It first casts a vote for the proposal, then tries to cast another vote for the same proposal, and expects the function to fail with an error message indicating that the voter has already cast their vote. The test then checks that the “totalVotesCast” counter has not been incremented.

One peculiarity of this test suite is that the voter registry is a private variable, so the tests cannot directly check whether a voter has already cast their vote. Instead, they expect the “doVote” function to fail if voters have already cast their vote. Another peculiarity is that the test suite does not check the final results of the voting, but only verifies that the votes have been cast. This is because the final result is only made publicly available once a proposal is in the Ended state.

```
1  describe("endVote function", () => {
2      it("should end a proposal's voting period and emit a VotingEnded
           event", async () => {
3        await ballotContract.endVote(proposalCount, { from:
             officialAddress });
4        const proposal = await ballotContract.proposals(proposalCount);
5        assert.equal(proposal.status, 2, "Proposal status should be
             Ended");
6        const counters = await ballotContract.helperCounters(
             proposalCount);
7        assert.equal(
8          counters.totalVotesInFavor,
9          1,
10         "Total votes in favor is incorrect"
11       );
12       const events = await ballotContract.getPastEvents("VotingEnded",
             {
13         fromBlock: 0,
14         toBlock: "latest",
15       });
16       assert.equal(
17         events.length,
18         1,
19         "VotingEnded event should have been emitted once"
```

```
20        );
21        assert.equal(
22          events[0].args.proposalId.toNumber(),
23          proposalCount,
24          "VotingEnded event has incorrect proposalId"
25        );
26        assert.equal(
27          events[0].args.votesInFavor.toNumber(),
28          1,
29          "VotingEnded event has incorrect result"
30        );
31      });
32
33      it("should revert if called by a non-official", async () => {
34        await ballotContract.createProposal("Proposal", "Description", {
35          from: officialAddress,
36        });
37        proposalCount = await ballotContract.proposalCounter();
38        await ballotContract.startVote(proposalCount, {
39          from: officialAddress,
40        });
41        try {
42          await ballotContract.endVote(proposalCount, { from:
              voterAddress });
43          assert.fail("Expected endVote to revert");
44        } catch (error) {
45          assert(
46            error.message.includes(
47              "Error: You are not the official who created this proposal
                  . Only the official can call this function"
48            ),
49            "Unexpected error message"
50          );
51        }
52        const proposal = await ballotContract.proposals(proposalCount);
53        assert.equal(
54          proposal.status,
55          1,
56          "Proposal status should not have changed"
57        );
58      });
59
60      it("should revert if called when the proposal is not in the Voting
          state", async () => {
```

```
61        await ballotContract.createProposal("Proposal", "Description", {
62          from: voterAddress,
63        });
64        proposalCount = await ballotContract.proposalCounter();
65        try {
66          await ballotContract.endVote(proposalCount, { from:
                officialAddress });
67          assert.fail("Expected endVote to revert");
68        } catch (error) {
69          assert(
70            error.message.includes(
71              "Error: The proposal is not in the state required to
                    perform this action"
72            ),
73            "Unexpected error message"
74          );
75        }
76        const proposal = await ballotContract.proposals(proposalCount);
77        assert.equal(
78          proposal.status,
79          0,
80          "Proposal status should not have changed"
81        );
82      });
83    });
```

This test suite is testing the functionality of the endVote function. The first test in the suite checks that the function correctly ends the voting period for a proposal and emits a `VotingEnded` event with the correct data. The second test ensures that the function reverts if called by someone other than the official who created the proposal. The third test verifies that the function reverts if called when the proposal is not in the Voting state. The test suite uses the assert function to check that the results of the function calls are as expected, and it uses the try...catch statement to catch any errors that may occur and check that the error messages are as expected.

5.4 Deployment

This section presents the deployment of the decentralized blockchain-based electronic voting platform for self-governance in organizations. The deployment consists of three main

components: frontend, blockchain setup, and smart contract deployment. The blockchain setup is the process of creating and configuring a private Ethereum network that hosts the voting platform. Smart contract deployment is the process of deploying the code that implements the voting logic and rules on the blockchain. Each component will be described in detail in this section, along with screenshots and code snippets to illustrate the steps involved.

The establishment of a private blockchain network is being considered to enable the completion of tasks that were unattainable with the ganache test network during the development of the smart contract.

First is an environment similar to production when many nodes are participating in the network; second, it provides a more controlled environment to test the contract's performance, security, and scalability. Furthermore, the private network is particularly suitable for the proposed voting system use case since it is likely that only members of the organization would be eligible to vote. Using a private network allows for more granular control over who has access to the network, ensuring that only authorized members can participate in the voting process. This enhances the security and privacy of the system, reducing the risk of unauthorized access and ensuring that the voting results are accurate and reliable.

5.4.1 Front-End Deployment

For the deployment of the front end, we created a build using Webpack with bundles all my assets and code into a single file the build is then served through an Express.js server on a port. This can be then uploaded to a cloud hosting platform, such as Netlify, to make it accessible to users via the Internet. Environment variables are set with the HTTP URL and port that is used to communicate with the blockchain node.

5.4.2 Private Blockchain Setup

In this project, we used Hyperledger Besu, an Ethereum client designed for businesses using Ethereum for public and private permissioned networks. Among its several consensus algorithms, we chose QBFT for our private blockchain network to serve as an electronic

voting platform due to its security, reliability, and performance advantages. QBFT is a federated consensus algorithm based on the Byzantine Fault Tolerance (BFT) algorithm that uses quorums within federations to confirm transactions and provide fast finality.

We designed a four-node blockchain, communicating virtually in the same machine using docker. The nodes communicate through a peer-to-peer network using the Ethereum wire protocol and the Devp2p protocol, allowing secure and reliable communication. Transactions are validated and added to the blockchain, with a malicious node needing to compromise multiple quorums to manipulate the election outcome, becoming more challenging with more nodes. In the case of an electronic voting blockchain, it may not be appropriate to charge voters for casting their votes this is the reason this private blockchain network has a 0 gas fee. In this regard, setting the gas fee to 0 ensures that voters can cast their votes without incurring any additional costs. This is crucial in an electronic voting blockchain as it should be accessible to everyone, regardless of their financial status. Additionally, eliminating the gas fee reduces the possibility of vote buying, a scenario where voters are paid to cast their votes for a particular candidate.

The steps to setup a private blockchain network as mentioned above is as follows. It is important to mention that the whole setup code is uploaded in a public repository on the appendix so it is easier to reproduce.

1. Setup a configuration file for the blockchain. In this configuration file, we can set up important properties of the blockchain such as the chain Id, the consensus algorithm, the gas fee, the complexity of the math puzzle, a nonce, the initial block count, some initially allocated accounts, and some other configurations.

2. Generate the genesis block file from the provided configuration file. This block will contain configurations of the blockchain.

3. Generate private and public key pairs for each one of the nodes that we are trying to set up. Copy these to each container. They will be used to start peer-to-peer communication in the blockchain.

4. Copy the genesis block file to each node container and create a node configuration file. In this node configuration file, we can set up the peer-to-peer networking con-

figuration, data storage location, HTTP network configuration, and logging settings.

5. With all nodes set up and running, the private blockchain network is ready to operate.

The private blockchain will start communicating with each node and interchange information about the nodes. They will connect and the network will be up and running, as can be seen in Fig. 5.5. The next step is to deploy our smart contract to the blockchain network.

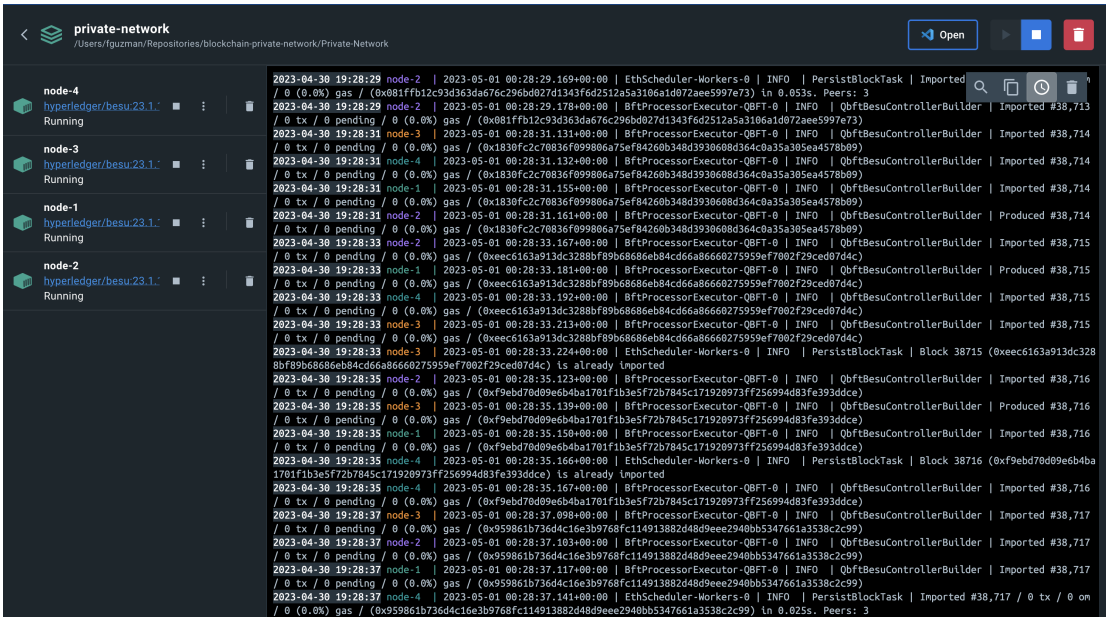


Figure 5.5: Private blockchain network running on Docker

5.4.3 Smart Contract Deployment

To accomplish this we can use the Truffle framework we used to develop the smart contract, we would just need to create an entry in the `truffle-config.js` file with the configuration of the private network. Mainly, the private key of one of the account nodes, a name for the network, and the network host as this network is local, *localhost* should suit us. Then we can run the command:

```
truffle migrate --<networkName>
```

This command will look for the *networkName* we described in the config file and pack the compilation of the smart contract into a transaction. Once deployed, it will allow us

to interact with the smart contract through one of the node's HTTP ports, which we set up on the node configuration files.

Chapter 6

Conclusions

This chapter provides an overview of the entire process involved in completing this thesis, beginning with a comprehensive literature review, continuing with the design and conceptualization of the system, followed by the development of a minimum valuable product, and concluding with testing and deployment of the application. The chapter culminates with an overall summary of the results and suggestions for future development.

6.1 Conclusions

The purpose of this project was to create a prototype of an electronic voting platform that effectively dealt with the issues of transparency, privacy, correctness, and integrity. To achieve this goal, we utilized agile methodologies that allowed us to develop a minimum viable product in a short period of time. The first step to creating such a platform was to review what was done in the literature about blockchain voting systems and what the main challenges and advantages were. Considering these factors, it was clear that the optimal choice for the blockchain network was Ethereum, as it boasts the largest community. This would ensure a wealth of available documentation, as well as potentially reach a wider audience with the electronic voting platform. Then began the process of creating the MVP of the platform, at the end of the development cycle the MVP provided a secure system that allows users to create and vote on proposals, all of which are deployed on a private blockchain that is accessible to every registered user on the platform. This approach effectively mitigated potential system failures, ensuring the reliability of the platform.

The use of blockchain technology was crucial to achieving specific goals that focused on

maintaining the integrity, privacy, and accessibility of the platform. This feature ensures that the system is tamper-proof and that every vote and proposal is immutable, allowing for a fair and secure election process. Furthermore, the development of a user-friendly interface played a vital role in ensuring accessibility, as it made the platform easy to use and facilitated the reading of results.

Developing such a platform is essential to promote fair and secure elections while empowering people to have more control over their representatives. This kind of platform encourages participation in the electorate, which is fundamental to the healthy functioning of any democracy.

In order to evidence that this thesis aligns with the proposed objective the following summary of the task performed for each objective is as the following:

1. To program an electronic decentralized voting platform using agile methodologies to create a minimum valuable product:
    - Documentation of requirements gathering and analysis process.
    - Implementation of an agile project management approach with clear sprints, user stories, and backlog management.
    - Deployment of a robust backend system, validated through unit testing, and integration testing.
    - Development of a user-friendly frontend interface to communicate with the backend
    - Delivery of a functional minimum viable product within the specified timeline, showcasing the successful implementation of agile methodologies.
2. To ensure the integrity, privacy, and reliability of the system using blockchain technology:
    - Research and select a suitable blockchain platform that aligns with the security requirements of the electronic voting system.
    - Design and implement smart contracts on the chosen blockchain platform to handle the voting process, ensuring immutability and transparency.

- Implement consensus mechanisms within the private blockchain network to ensure the reliability and validity of the voting results.
- Documenting the implemented blockchain technology, including details of the selected platform, smart contracts, and consensus mechanisms employed.

3. To guarantee the accessibility of the system:

   - Designing an intuitive and visually appealing (GUI) that accommodates various accessibility requirements, such as color contrast, font size, and screen reader compatibility.
   - Test usability by using heuristic evaluation of the user interface. While it does not replace user testing it sets the ground for focusing usability first.
   - Implementing user-friendly navigation and interaction patterns within the GUI to ensure ease of use and seamless navigation.
   - Incorporating accessibility features, such as alternative text for images, keyboard navigation support, and adjustable text size options.

4. To mitigate possible system failure:

   - Selecting a reliable and scalable web server infrastructure that meets the performance requirements of the electronic voting platform.
   - Configuring the web server to host the electronic voting web application securely.
   - Comprehensive documentation of the web server configuration and private blockchain network configuration

Although there remains much work to be done, this project lays the foundation for the development of blockchain applications with a focus on purposes other than finances. This project offers a blueprint for future endeavors in this field of governability, serving as a guide for those looking to achieve similar objectives. With this technology at our disposal, it is crucial to shift our focus toward prioritizing the interests of people and improving their quality of life.

6.2 Future Work

This section outlines several potential areas for improvement in the blockchain e-voting platform, which could enhance the system. These areas include security enhancements, accessibility enhancements, usability enhancements, and technical enhancements.

Security Enhancements

We can implement two-factor authentication to comply with the latest security standards. This would require voters to provide two forms of identification, such as a password and a unique code sent to their phone or their preferred vault application, before accessing the platform. This would significantly increase the platform's security and protect against fraudulent voting. Additionally, conducting regular security audits is necessary to identify and fix any vulnerabilities in the system. To accomplish this, we will need to set up an auditor role with access to the blockchain logs to check that system is working as expected.

Accessibility Enhancements

To make the platform accessible to all voters, it is recommended to introduce an accessibility feature for disabled voters. This could include providing alternative options for voters who are visually or hearing impaired, such as audio or visual aids. Developing a mobile application would also improve accessibility, allowing voters to access the platform easily and conveniently from their mobile devices. This would increase the platform's accessibility and make it more user-friendly for voters who prefer mobile devices.

One important aspect to improve is how a user can add himself as part of the network, right now the process consists of a user getting his private key from the system administrator and setting up the node. In the future, we would like to use Docker Compose to set up nodes on a private network. They could simply download the Docker Compose file and run the command to set up their own node. This would allow new users to join the network quickly and easily, without requiring special technical knowledge or assistance.

Usability Enhancements

To improve the user experience, it is recommended to provide a user-friendly interface for easy navigation. This would make the platform more intuitive and easy to use, allowing voters to easily navigate and understand the platform's features.

Creating manuals to easily create a private blockchain network with its nodes would help increase the adoption and use of the e-voting platform. This would allow more organizations and institutions to establish private blockchain networks and integrate the e-voting platform into their existing systems.

Technical Enhancements

Introducing biometric verification for voter identification would also improve security and prevent fraudulent voting. This would involve using biometric data such as fingerprints or facial recognition to verify a voter's identity before they can access the platform.

Adding a maintainer role that can approve and review a proposal and allowing users to flag proposals for maintainers to review would also enhance usability. This would improve the quality of proposals and prevent inappropriate or fraudulent proposals from being approved.

Furthermore, allowing users to make suggestions to improve a proposal, which can gather votes to improve the proposal, would increase voter engagement and improve the quality of proposals. A time-out function could also be created to end a proposal instead of manually ending the voting period, increasing security and usability by ensuring that voting ends at a predetermined time.

Bibliography

[1] J. Brennan, *Against Democracy*. Princeton: Princeton University Press, 2017. [Online]. Available: https://doi.org/10.1515/9781400888399

[2] D. Altman, *Direct democracy worldwide*. Cambridge University Press, 2010.

[3] P. G. Neumann, "Security criteria for electronic voting," in *16th National Computer Security Conference*, vol. 29, 1993, pp. 478–481.

[4] P. C. Schmitter and T. L. Karl, "What democracy is... and is not," *Journal of democracy*, vol. 2, no. 3, pp. 75–88, 1991.

[5] K. M. Khan, J. Arshad, and M. M. Khan, "Secure digital voting system based on blockchain technology," *International Journal of Electronic Government Research (IJEGR)*, vol. 14, no. 1, pp. 53–62, 2018.

[6] A. Mugica, "The case for election technology," *European View*, vol. 14, no. 1, pp. 111–119, 2015. [Online]. Available: https://doi.org/10.1007/s12290-015-0355-5

[7] M. M. Sarker and T. M. N. U. Akhund, "The roadmap to the electronic voting system development: a literature review," *International Journal of Advanced Engineering, Management and Science*, vol. 2, no. 5, p. 239465, 2016.

[8] J. P. Gibson, R. Krimmer, V. Teague, and J. Pomares, "A review of e-voting: the past, present and future," *Annals of Telecommunications*, vol. 71, no. 7, pp. 279–286, 2016.

[9] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[10] H. Wu, "The hash function jh," *Submission to NIST (round 3)*, vol. 6, 2011.

[11] U. Jafar, M. J. A. Aziz, and Z. Shukur, "Blockchain for electronic voting system—review and open research challenges," *Sensors*, vol. 21, no. 17, p. 5874, 2021.

[12] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the works of leslie lamport*, 2019, pp. 203–226.

[13] E. Kapengut and B. Mizrach, "An event study of the ethereum transition to proof-of-stake," *arXiv preprint arXiv:2210.13655*, 2022.

[14] S. Joshi, "Feasibility of proof of authority as a consensus protocol model," *arXiv preprint arXiv:2109.02480*, 2021.

[15] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 2567–2572.

[16] H. Sheth and J. Dattani, "Overview of blockchain technology," *Asian Journal For Convergence In Technology (AJCT) ISSN -2350-1146*, Apr. 2019. [Online]. Available: https://asianssr.org/index.php/ajct/article/view/728

[17] S. Tikhomirov, "Ethereum: state of knowledge and research perspectives," in *International Symposium on Foundations and Practice of Security*. Springer, 2017, pp. 206–221.

[18] D. Vujičić, D. Jagodić, and S. Ranđić, "Blockchain technology, bitcoin, and ethereum: A brief overview," in *2018 17th international symposium infoteh-jahorina (infoteh)*. IEEE, 2018, pp. 1–6.

[19] C. Feng and J. Niu, "Selfish mining in ethereum," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1306–1316.

[20] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[21] S. Rouhani and R. Deters, "Performance analysis of ethereum transactions in private blockchain," in *2017 8th IEEE international conference on software engineering and service science (ICSESS)*. IEEE, 2017, pp. 70–74.

[22] "Ethereum wallets," https://ethereum.org/en/wallets/, accessed: 2022-11-04.

[23] K. Wu, Y. Ma, G. Huang, and X. Liu, "A first look at blockchain-based decentralized applications," *Software: Practice and Experience*, vol. 51, no. 10, pp. 2033–2050, 2021.

[24] M. McCormick, "Waterfall vs. agile methodology," *MPCS, N/A*, vol. 3, 2012.

[25] M. Fowler, J. Highsmith *et al.*, "The agile manifesto," *Software development*, vol. 9, no. 8, pp. 28–35, 2001.

[26] A. Srivastava, S. Bhardwaj, and S. Saraswat, "Scrum model for agile methodology," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 2017, pp. 864–869.

[27] K. Schwaber and M. Beedle, *Agile software development with scrum. Series in agile software development*. Prentice Hall Upper Saddle River, 2002, vol. 1.

[28] E. Ries, "Minimum viable product: a guide," *Startup lessons learned*, vol. 3, p. 1, 2009.

[29] "Front end vs back end of your website: Everything you need to know," https://lets godojo.com/front-end-vs-back-end/, accessed: 2022-11-04.

[30] F. Wessling and V. Gruhn, "Engineering software architectures of blockchain-oriented applications," in *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2018, pp. 45–46.

[31] R. Hanifatunnisa and B. Rahardjo, "Blockchain based e-voting recording system design," in *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*. IEEE, 2017, pp. 1–6.

[32] J.-H. Hsiao, R. Tso, C.-M. Chen, and M.-E. Wu, "Decentralized e-voting systems based on the blockchain technology," in *International Conference on Ubiquitous Information*

Technologies and Applications, International Conference on Computer Science and its Applications. Springer, 2018, pp. 305–309.

[33] S. A. Adeshina and A. Ojo, "Maintaining voting integrity using blockchain," in *2019 15th International Conference on Electronics, Computer and Computation (ICECCO)*. IEEE, 2019, pp. 1–5.

[34] X. Yang, X. Yi, S. Nepal, and F. Han, "Decentralized voting: A self-tallying voting system using a smart contract on the ethereum blockchain," in *Web Information Systems Engineering – WISE 2018*, H. Hacid, W. Cellary, H. Wang, H.-Y. Paik, and R. Zhou, Eds. Cham: Springer International Publishing, 2018, pp. 18–35.

[35] A. Young and S. Verhulst, "Self sovereign identity for government services in zug, switzerland," 2018.

[36] N. Naik and P. Jenkins, "uport open-source identity management system: An assessment of self-sovereign identity and user-centric data platform built on blockchain," in *2020 IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, 2020, pp. 1–7.

[37] Agora, "Agora: Bringing our voting systems into the 21st century [whitepaper version 0.2]," Tech. Rep., 2020. [Online]. Available: https://static1.squarespace.com/static/5b0be2f4e2ccd12e7e8a9be9/t/5f37eed8cedac41642edb534/1597501378925/Agora_Whitepaper.pdf

[38] R. Bulut, A. Kantarcı, S. Keskin, and Ş. Bahtiyar, "Blockchain-based electronic voting system for elections in turkey," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*. IEEE, 2019, pp. 183–188.

[39] S. Anjan and J. P. Sequeira, "Blockchain based e-voting system for india using uidai's aadhaar," *Journal of Computer Science Engineering and Software Testing*, vol. 5, no. 3, pp. 26–32, 2019.

[40] Y. M. Wahab, A. Ghazi, A. Al-Dawoodi, M. Alisawi, S. S. Abdullah, L. Hammood, and A. Y. Nawaf, "A framework for blockchain based e-voting system for iraq." *International Journal of Interactive Mobile Technologies*, vol. 16, no. 10, 2022.

[41] J. Budurushi, R. Jöris, and M. Volkamer, "Implementing and evaluating a software-independent voting system for polling station elections," *Journal of information security and applications*, vol. 19, no. 2, pp. 105–114, 2014.

[42] K. Teja, M. Shravani, C. Y. Simha, and M. R. Kounte, "Secured voting through blockchain technology," in *2019 3rd international conference on trends in electronics and informatics (ICOEI)*. IEEE, 2019, pp. 1416–1419.

[43] M. Swan, *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc.", 2015.

[44] F. Hjálmarsson, G. K. Hreiarsson, M. Hamdaqa, and G. Hjálmtýsson, "Blockchain-based e-voting system," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018, pp. 983–986.

[45] G. Keirns, "Local government in south korea taps blockchain for community vote," Sep 2021. [Online]. Available: https://www.coindesk.com/markets/2017/03/07/local-government-in-south-korea-taps-blockchain-for-community-vote/

[46] R. Taş and Tanrıöver, "A systematic review of challenges and opportunities of blockchain for e-voting," *Symmetry*, vol. 12, no. 8, 2020. [Online]. Available: https://www.mdpi.com/2073-8994/12/8/1328

[47] S. T. Alvi, M. N. Uddin, and L. Islam, "Digital voting: A blockchain-based e-voting system using biohash and smart contract," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2020, pp. 228–233.

[48] A. Alam, S. M. Zia Ur Rashid, M. Abdus Salam, and A. Islam, "Towards blockchain-based e-voting system," in *2018 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, 2018, pp. 351–354.

[49] M. Pawlak, J. Guziur, and A. Poniszewska-Marańda, "Voting process with blockchain technology: auditable blockchain voting system," in *International Conference on Intelligent Networking and Collaborative Systems*. Springer, 2019, pp. 233–244.

[50] G. Sun, M. Dai, J. Sun, and H. Yu, "Voting-based decentralized consensus design for improving the efficiency and security of consortium blockchain," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6257–6272, 2021.

[51] G. Xu, Y. Liu, and P. W. Khan, "Improvement of the dpos consensus mechanism in blockchain based on vague sets," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4252–4259, 2019.

[52] Y. Liu, G. Wang, and L. Feng, "A general model for transforming vague sets into fuzzy sets," *Transactions on computational science II*, pp. 133–144, 2008.

[53] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Financial Cryptography and Data Security*, A. Kiayias, Ed. Cham: Springer International Publishing, 2017, pp. 357–375.

[54] S. Dhir, D. Kumar, and V. Singh, "Requirement paradigms to implement the software projects in agile development using analytical hierarchy process," *International Journal of Decision Support System Technology (IJDSST)*, vol. 9, no. 3, pp. 28–41, 2017.

[55] B. Crosby, *Stakeholder analysis: a vital tool for strategic managers.* USAID's Implementing Policy Change Project, 1992.

[56] R. Malan, D. Bredemeyer *et al.*, "Functional requirements and use cases," *Bredemeyer Consulting*, 2001.

[57] L. Marchesi, M. Marchesi, and R. Tonelli, "Abcde –agile block chain dapp engineering," *Blockchain: Research and Applications*, vol. 1, p. 100002, 12 2020.

[58] D. Spinellis, "Git," *IEEE software*, vol. 29, no. 3, pp. 100–101, 2012.

[59] "What is truffle?" [Online]. Available: https://trufflesuite.com/docs/truffle/

[60] Trufflesuite, "Trufflesuite/ganache: A tool for creating a local blockchain for fast ethereum development." [Online]. Available: https://github.com/trufflesuite/ganache#readme

[61] "Getting started with metamask." [Online]. Available: https://metamask.zendesk.com/hc/en-us/articles/360015489531-Getting-Started-With-MetaMask

[62] C. Gackenheimer and C. Gackenheimer, "What is react?" *Introduction to React*, pp. 1–20, 2015.

[63] "Introducing jsx." [Online]. Available: https://reactjs.org/docs/introducing-jsx.html

[64] G. Bierman, M. Abadi, and M. Torgersen, "Understanding typescript," in *ECOOP 2014 – Object-Oriented Programming*, R. Jones, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 257–281.

[65] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1990, pp. 249–256.

[66] B. B. Bederson, B. Lee, R. M. Sherman, P. S. Herrnson, and R. G. Niemi, "Electronic voting system usability issues," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2003, pp. 145–152.

[67] D. Arnyndiasari, R. Ferdiana, and P. I. Santosa, "Software practices for agile developers: A systematic literature review," in *2022 1st International Conference on Information System Information Technology (ICISIT)*, 2022, pp. 238–243.

Appendices

.1 Appendix 1. Code Repository for the Electronic Voting System

The code repository for the electronic voting system can be accessed at the following URL: `https://github.com/FranzGB/e-voting-contract`.

.2 Appendix 2. Code Repository for the Private Blockchain Network

The code repository for the private blockchain network can be found at the following URL: `https://github.com/FranzGB/blockchain-private-network`.

.3 Appendix 3. Project Management Utilities

The project management utilities utilized for this project include the following:

Kanban Board: The Kanban board for tracking project tasks and progress can be accessed at: `https://trello.com/b/Ylwqyy5r/electronic-voting-platform`.

Backlog: The project backlog, which lists pending tasks and requirements, can be accessed at the following URL: `https://docs.google.com/spreadsheets/d/1H48_z9gMDu6WoTyKcgeuYV_3N7bDXQSiROiN3gHspu8/edit?usp=sharing`.