CA400 4th Year Project

Vot.ie - A decentralised blockchain based voting system

Contributors:
Bartlomiej Kiraga - 17327333
Rónán Mac Gabhann - 17478562

Supervisor:
Dr. Geoffrey Hamilton

Section 1: Introduction

1.1 Overview

Our proposed project is a web-app voting platform built on Ethereum to provide a secure, decentralised and incorruptible platform for voting.

Features Include:

- An authentication system.
- Voter education, eg guides.
- Transparency, guide to verifying information on the vote.
- The ability to cast your vote.
- Live voting statistics.
- A decentralised Ethereum network composed of all voters maintaining a secure verifiable ledger.
- A website to access all above features.

The current political climate could greatly benefit from a secure decentralised voting. The onset of the COVID-19 pandemic means that there is an increased demand for ways to vote remotely and securely, our application will provide a means to vote remotely while ensuring election integrity, a concern that has appeared since remote voting has become mainstream.

Our app would be used in tandem with voting centres for those without access to the internet or computers themselves, at voting centres computers would be provided, or computers which can read a paper vote as long as you're on the electoral register.

1.2 Business Context

We intend for our application to be used in democratic political elections across the free world. Elections can be a costly process when you account for the ballot costs, the operating costs of the voting stations and the human work hours required to examine and count all submitted ballots. These costs could be greatly reduced with a widespread adoption of an online voting system that could remove or simplify many of the processes needed in normal elections.

1.3 Glossary

Blockchain: Decentralized distributed ledger of records represented as cryptographically linked blocks, designed to prevent data from being retroactively altered.

Ethereum: Decentralized open-source blockchain platform designed to incorporate smart contracts.

Smart Contracts: Program which is intended to automatically execute an event or action according to terms of a pre-decided contract.

Solidity: Programming language used for implementing smart contracts on blockchain platforms such as Ethereum.

Public ledger: A decentralised record keeping system, used to anonymously document votes. Shared among all voters, their computers work together to verify the vote's cast.

Electoral register: A list of eligible, registered voters maintained by the relevant public services.

Web3: The decentralised web standard, appears as regular web pages but operates on a decentralised network such as Ethereum.

Section 2: General Description

2.1 Product / System Functions

2.1.1 Start an election

One of the critical functions of the system is to allow users to create an election. To do this users are provided with an interface which allows them to set up election parameters such as the running candidates and the timeline of the election. Any user can start their own election instance, someone who starts an election instance becomes an administrator for that election.

2.1.2 Cast a vote

Another critical function of the system is to allow users to cast a vote. The user interface allows voters to cast their ballots with their unique identification number such as PPSN attached. The system then checks the validity of the ballot and whether the user is an authorized voter in the election. If all checks pass the ballot is counted.

2.1.3 Tally votes

After the election is over the votes must be counted in order to declare a winner. The tallying process is designed to be decentralized as everyone can view the blockchain and can tally the votes themselves, providing an additional layer of transparency to the election process.

2.2 Primary Actors

2.2.1 Administrator

Administrators set election parameters before voting begins.

1. Setting the election instance's voting system.
2. Setting the election instance's start and end dates.
3. Adding additional administrators to the system.
4. Setting election variables, eg. Can users view live voting statistics for this election instance?
5. Setting the election instance's ballot, such as the nominees/parties listed.

Administrator accounts are not a part of the network, their contribution takes place offline and they cannot change election parameters once the election instance is live.

2.2.2 Voter

Voters would be any user on the electoral register.

1. Casting your vote, adding it to the public ledger.
2. Verifying other's votes on the public ledger, this is a mandatory and automatic process done when you cast your own vote.

Voters also have access to all system functions of a User.

2.2.3 User

A user is anyone who connects to the network. They may or may not be a registered voter, however they still have access to some system functions.

1. View live voting statistics, if enabled by the administrator pre-voting.
2. View the election information such as the nominees.
3. View other documents shared by administrators, such as guides to voting and how to register.

2.3 Secondary Actors

2.3.1 Ethereum network

The Ethereum network hosts the various contracts and web3 pages created by the Administrator in the election builder. The users will connect to the Ethereum network to vote and view statistics.

2.3.2 Encryption engine

The encryption engine generates our keys and our blinds to ensure anonymity among voters.

2.3.3 Election builder

Provides an interface for the administrator to set parameters for the election such as candidates running and the timeline of the election, they're choices will create a simple variables file which the migration contract will read to deploy the relevant contracts and web files.

2.3.4 Migration Contract

Deploys all the contracts necessary for the election.

2.3.5 Voting contract

Handles the casting of anonymised ballots and stores them in the blockchain after ensuring that they were signed by the validation contract.

2.3.6 Validation contract

Is sent the blinded hash of a voter's ballot and verifies that the voter is eligible to vote. If it is it will sign the blinded hash and return it to the voter.

2.3.7 Web3 page

Ethereum's API that connects the Nodejs application to the blockchain. Allows the hosting of decentralized websites on the Ethereum blockchain.

2.4 User Characteristics and Objectives

The main target users for our application are people participating in government elections, however its use could be expanded to other smaller scale voting events that would require security or transparency of votes. Any user with an internet connection should be able to use our application without any prior knowledge. If a user does not have access to the internet they can use public computers in polling stations.

The main objective for our voting application is to decentralize the election procedure in a way that maintains its security and ease of use. Currently a voter has to trust the integrity of the government institution to count their vote fairly.
By incorporating blockchain technology our application aims to give its users transparency regarding their elections and a peace of mind that their vote has been counted fairly. Regardless whether an election is fair or not, our application has the potential to ease public unrest and prevent corruption during the voting process.

Ease of use is another objective for our application. Being able to vote remotely online is an important benefit for people with busy schedules or accessibility issues which would have the benefit of raising voter turnout.

2.5 Operational Scenarios

2.5.1 Use Case: Creating an Election Instance

CHARACTERISTIC INFORMATION
Goal in Context: To create an election instance on the blockchain.
Scope: User client, blockchain contract.
Level: Primary task
Preconditions: Admin has downloaded the application files.
Success End Condition: The election instance is initialised containing candidate information and the election end date. Authorised users may participate in the election.
Failed End Condition: The election fails to initialize.
Primary Actor: Admin
Trigger: Admin selects the create an election option.

MAIN SUCCESS SCENARIO

1. Admin sets election parameters including candidates, the timeline of the election and the list of authorized users that may participate.
2. Admin initializes the election.
3. The election instance is now hosted on the Ethereum network

RELATED INFORMATION
Priority: Top priority
Frequency: Low, once at the start of every election
Channel to primary actor: Local machine and blockchain network

2.5.2 Use Case: Casting a Vote

CHARACTERISTIC INFORMATION

Goal in Context: To submit a ballot anonymously to the blockchain.
Scope: User client, encryption engine and blockchain contract.
Level: Primary task
Preconditions: Voter is registered.
Success End Condition: Ballot is cast anonymously by a registered voter.
Failed End Condition: Ballot is not recorded, or ballot from an unregistered voter is recorded.
Primary Actor: Voter
Trigger: Submitting a ballot through the eth3 webpage.

MAIN SUCCESS SCENARIO

1. A user connects to our web3 page.
2. The voter fills out a ballot client side and fills in their personal information.
3. The ballot is hashed and blinded and sent to a blockchain contract which signs it if they are eligible to vote.
4. In an automated process, the user client removes their personal information and unblinds their signed ballot.
5. The user client anonymously adds their ballot to the blockchain, sending the signed hash of their vote as proof they are eligible.

RELATED INFORMATION

Priority: Top priority.
Performance Target: Web3 page loading in < 3 seconds, vote authenticated and cast to blockchain in < 10seconds.
Frequency: High, once per voter.
Secondary Actors: Inspector (if manually checking electoral register)

2.6 Constraints

2.6.1 Ethereum Blockchain

All election instances would be reliant on the Ethereum blockchain and any performance constraints associated with it. All blocks added to the blockchain have to be validated by miners, so total performance depends on the combined miners performance and the current load of the blockchain, this may mean that votes will be queued before being added to the blockchain.
Also, Ethereum has changed it's modus operandi in the past when it branched into Ethereum Classic and Ethereum, so it is possible future major overhauls to the Ethereum network could impact our application.

2.6.2 Electoral Systems

To be widely adopted, our application would have to comply with the various standards, regulations and voting methodologies used in different elections and or regions of the world. Universal democratic standards include accessibility for all voters, ensuring the anonymity of voters and maintaining security such that the election cannot be tampered with and votes are tallied accurately.

2.6.3 Timeline

Development of the project has begun as of the 20th of November 2020 and must be delivered in full on the 7th of May 2021.

Section 3: Functional Requirements

3.1 Election Builder

Description:

The election builder allows a user to set up an election. The user inputs parameters such as the candidate details, a list of authorized voters and the timeline of the election, which results in creation of an election instance. The user becomes the admin of the election and other authorized users may vote in the election.

Criticality:

Critical, there is a need for a way to initialize the election and give it all their required parameters. All elections require a start and end date as well as information regarding candidates who participate.

Technical issues:

Allowing for multiple different parameters to be adjusted by the administrator, while ensuring all election formats are fully functional and bug free. The design should be intuitive so it is clear what the final election instance will look like.

Dependencies with other requirements:

The contracts deployed must be made in advance

3.2 Voting Contract

Description:

The voting contract will be held on the Ethereum network and will be called by all voters when casting their ballots, it will add their vote to the tally if they are an eligible voter as checked by the validation contract

Criticality:

Critical, the voting contract is the core functionality of our system and without it elections couldn’t be hosted.

Technical issues:

Multiple variations of the voting contract must be written in Solidity to allow for different voting methodologies and parameters as set by the Administrator.

Dependencies with other requirements:

This is one of the contracts that will be deployed by the administrator in the election builder. It checks for a signature from the validation contract to ensure that the voter is eligible to vote.

It can interact with the statistics engine to create semi-live statistics of the current tally.

3.3 Validation Contract

Description:

The validation contract is sent a blinded hash of a voter's ballot and checks if that voter is eligible to vote, if they are it will sign the ballot and return it to the voter, who can then submit they're anonymous unencrypted ballot along with their signed hashed vote to the voting contract to add it to the blockchain. This whole exchange is automated after the user casts their ballot.

Criticality:

High, it is vital to ensure only registered voters can vote and in maintaining voter's anonymity.

Technical issues:

To ensure the eligibility of voters it must be given access to the electoral register in a private manner, as it is not publicly available information.

Dependencies with other requirements:

Only votes accompanied by a signature from the validation contract will be accepted by the voting contract.

3.4 Web3 Site

Description:

The Web3 site is a decentralised website hosted on the Ethereum network, users can connect to it through several different applications, such as Metamask. Our site will host the interface for both the voter and the unregistered user. The voter can cast their ballots, view voting guides, and check the live statistics. The user can view the voting guides and live statistics but cannot vote.

The site will be fully accessible and comply with the web content accessibility guidelines to allow the visually impaired to vote.

Criticality:

Medium. While a user interface isn't strictly required for a voter to cast their vote, the majority of people lack the tech know-how to deliver their vote to the network themselves and it would be unreasonable to expect a society to do so.

Technical issues:

Ensuring full accessibility as per the Web Content Accessibility Guidelines.

Ensuring a user can register an Ethereum wallet easily here, as most users won't have one.

Dependencies with other requirements:

The site will be the interface to our system, hosting our smart contracts, statistics page and voter guide.

3.5 Statistics Engine

Description:

The voting application includes a way to gather voting statistics. Instead of waiting until the end of the election the statistics engine allows users to view current tally results before all votes have been cast.

Criticality:

Low, the purpose of the statistics feature is to improve user experience rather than a core functionality of the election.

Technical issues:

The release of the tally information has to be timed in a way to prevent the discovery of the identity of a ballot caster through the use of a timing attack.

Dependencies with other requirements:

Relies on the voting contract for the current tally.

Relies on the Web3 site to display the statistics.

3.6 Voter Guide

Description:

The user interface will contain a simple guide about the use of our application.

Criticality:

Low, the user guide is meant to improve the user experience and is not important to the functionality of the system.

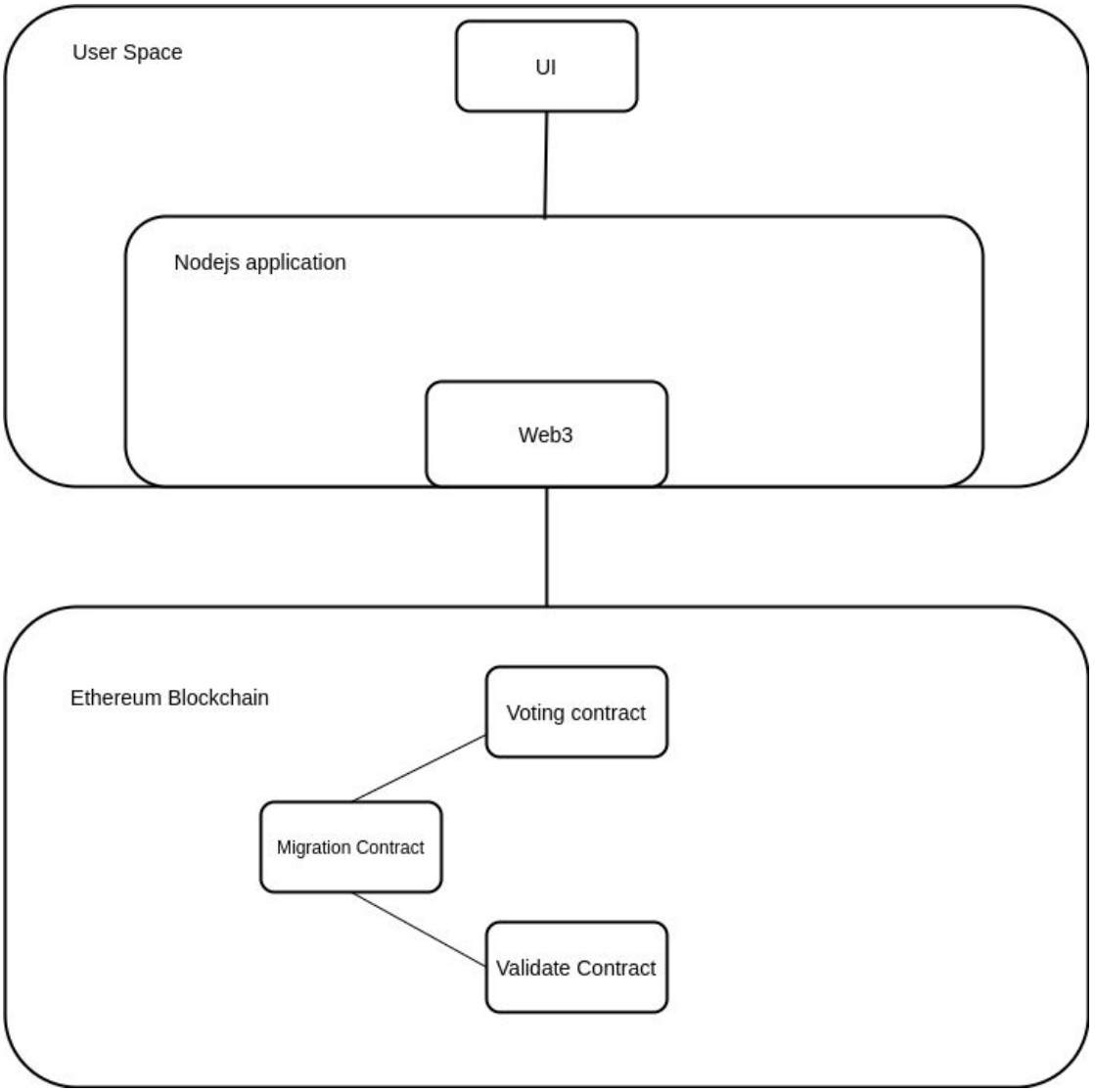
Technical issues:

The guide has to be written in a way that describes our application in simple non-technical language that could be understood by any user.

Dependencies with other requirements:

Requires a working Web3 user interface.

Section 4: System Architecture



The client side of the system is a Nodejs application which connects to the Ethereum blockchain using the Web3 API. The Voting and Validate contracts required for the functionality of the election are deployed onto the blockchain using the Migration contract.

Section 5: High Level Design

5.1 Election Builder

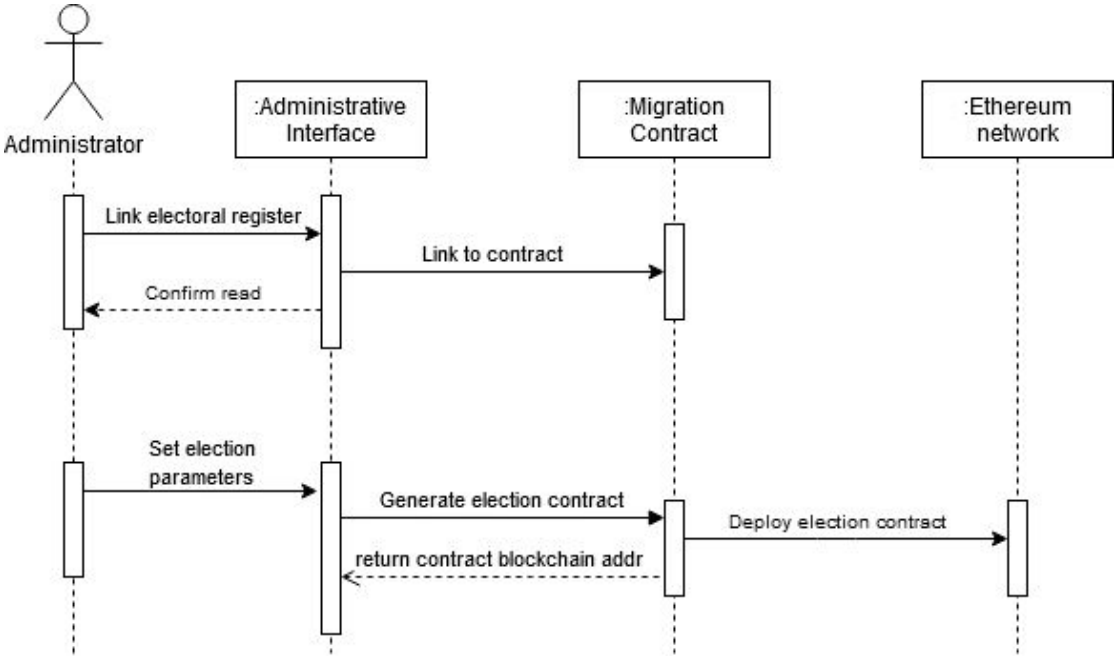


Figure 5.1 shows a sequence diagram of the election builder in action. First of all, the administrator must link the electoral register into the system through the migration contract. This will allow the verification contract to check if voters are present on the register and therefore eligible to vote.

The administrator then sets the election parameters, which through the election builder generates the appropriate election contract for his choices (eg. voting methodology, whether statistics are viewable, timeframe). The election contract is then deployed to the Ethereum network where it can be used by all when voting begins. The blockchain address is returned to the administrator so he knows where it is being hosted.

The election is now hosted on the Ethereum network and can be interfaced through a Web 3 client.

5.2 Cast Vote

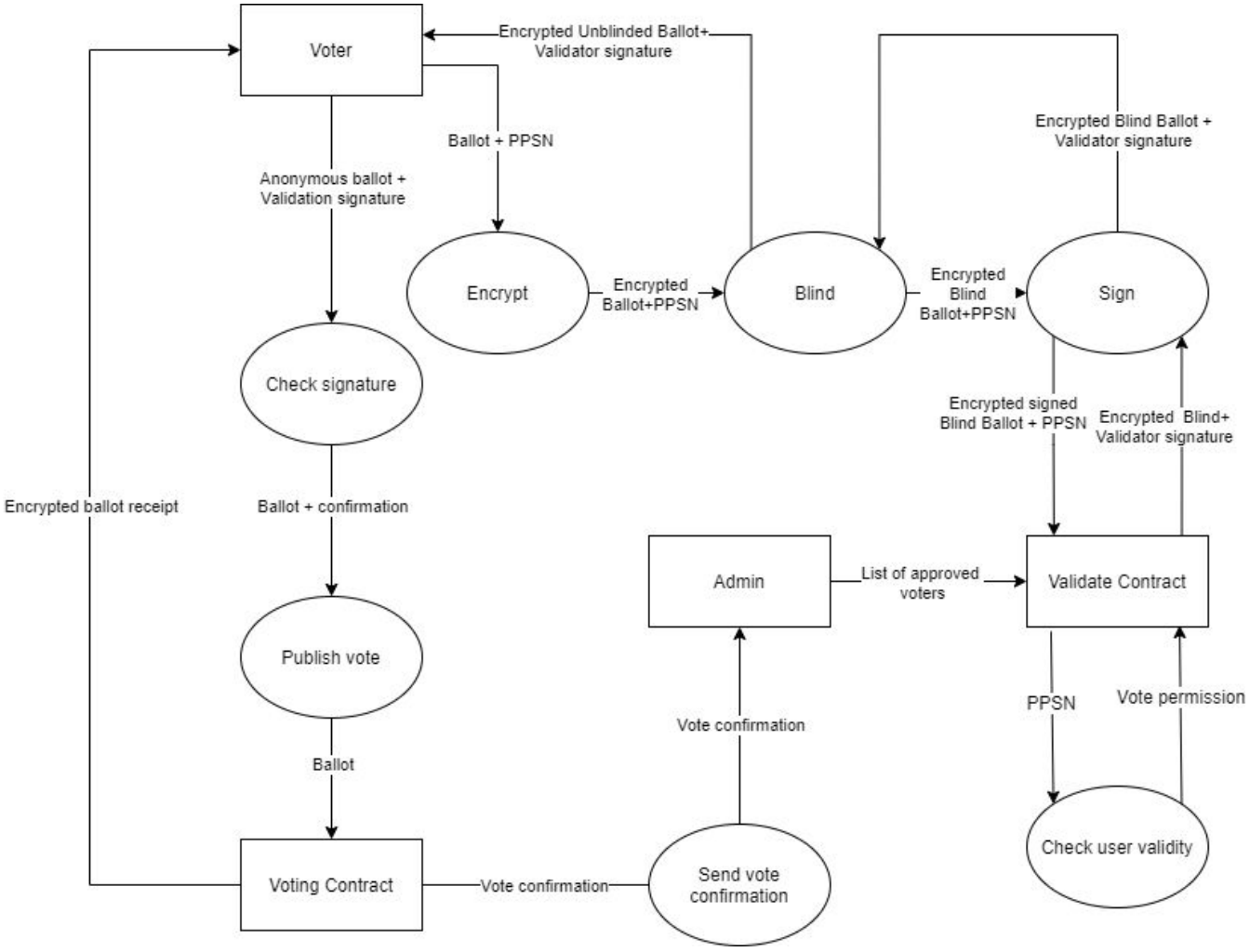
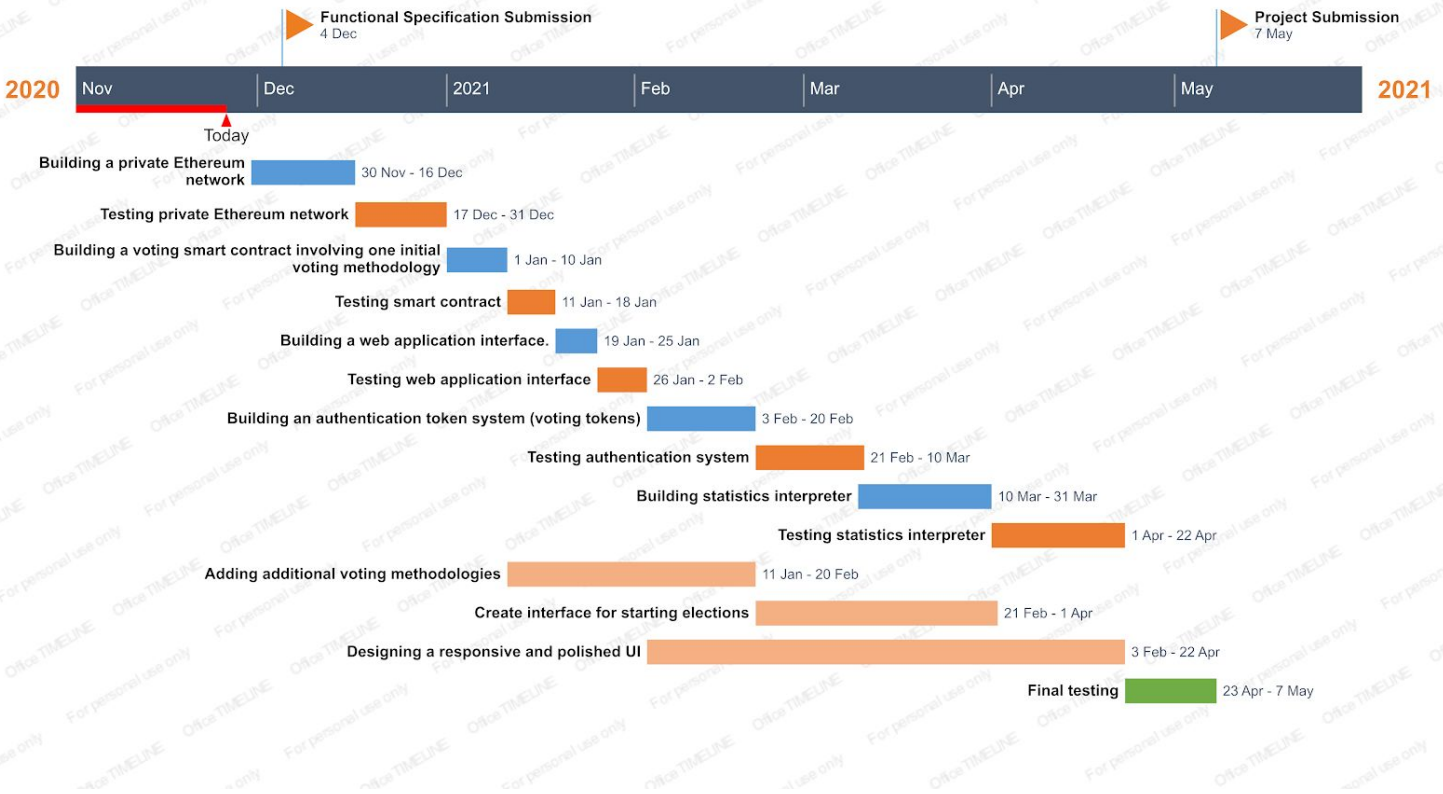


Figure 5.2 shows the process involved in casting a vote.
The voting casting process uses blind signatures to verify and anonymize valid voters and their ballots.

When a user casts a vote their ballot along with their ID (PPSN in case of national elections) gets hashed, blinded and goes through a validator contract which checks the voter's information. If the voter is on the electoral register, it is returned with the validator's signature. The voter then removes the blinding encryption layer which leaves behind a hashed ballot with the validator's signature.
The ballot is then submitted to the blockchain along with the hashed signed ballot, if the hashed signed ballot has a valid signature it is accepted and added to the blockchain.

Section 6: Preliminary Schedule



7. References

1. Ethereum: https://ethereum.org/en/
2. Solidity: https://github.com/ethereum/solidity
3. Web Content Accessibility Guidelines 2.1: https://www.w3.org/TR/WCAG21/
4. Metamask: https://metamask.io/