Decentralisation of polling processes in developing nations: VoteBlocks Project Report

Sumuk Shashidhar - 12A
Adithya Mahesh -12A

Sri Kumaran Children's Home

March 16, 2021

Abstract

We put forward an idea and a prototype for an open-source, low-power, polling system built on traditional blockchain technology to decentralise the election process in developing nations in order to protect democracy and prevent voter fraud. Simple resource-efficient hashing mechanisms are used in order to ensure energy-efficiency but at the same time rely on the safety in scale approach to preventing attacks in the electoral process. Limitations of the voter-registration process and partial decentralisation problems are carefully examined and solutions are proposed for the same.

Keywords— elections, democracy, blockchain, decentralised, crypto

Sri Kumaran Children's Home, CBSE

Bangalore

[redacted]

This is to certify that

……………………..........................

Of Class XII has successfully completed the project work titled --- under the guidance of -----------------------------------during the academic year ____________in partial fulfillment of ______________practical examination conducted by AISSCE.

________________________ ________________________

Internal Examiner ***External Examiner***

____

Principal

Date of Practical Examination : _________________

Name of the Candidate: ______________________________

Registration Number: ______________________________

Examination Centre: ______________________________

Acknowledgements

The success of a project depends upon the persistent of an individual and the sustained support received from a few other who are equally responsible for their precious appreciation of such endeavours. My strength is all due to my honourable Principal Deepa Sridhar, who has been an unending source of inspiration and support towards accomplishment of this project.

I would like to express my deepest sense of gratitude towards my computer science teacher Ms. Pavani Kanukollu without whom I could not have successfully completed this project.

I would also thank all my friends who helped me create such a project.

My personal gratitude is extended towards my parents, who have been a constant source of encouragement and support in the sense of this project. Last but not least I want to thank Almighty for enlightening, strengthening and guiding me in the completion of the project.

Contents

1 Introduction

Democracy in developing nations is often challenged by electoral opacity. A lack of technological advancement/accessibility is a crutch and contributes to widespread voter fraud and voter suppression [1]. To combat this, we propose a decentralised plug-and-play polling system that allows the general public to take part in both aspects of the electoral process: organisation and voting. This also plays a psychological role as it allows the public to personally invest themselves in the process, thereby increasing voter turnouts and national democratic trust.

VoteBlocks is the proposed prototype that is built on Python and Flask to ensure compatibility with a wide range of available hardware and minimal dependencies [2]. It consists of two parts that can be hot swapped: a frontend web application and a backend server. These can be run on the same machine or independently, thereby ensuring ease of use and greater control. It follows a traditional blockchain approach to recording of anonymous votes and allows any independent party to participate in the electoral process.

1.1 The need

VoteBlock is a decentralized, trustless voting system built using the concepts of Blockchain. It digitalizes a very important aspect of a democracy's needs - the receiving, storage and counting of votes in elections. It also addresses the skepticism of security of the same while doing so, as it doesn't rely on the trust of one single centralized entity which can have control over everything.

The storage of votes as "hashes" and the decentralized yet consistent nature in which the votes are stored, together ensure the authenticity of the votes and thereby the election results.

The world is moving towards the use of technology to simplify more and more needs. On the other hand, the concerns of security in this matter are on the rise too. This application embraces the use of technology in the process of democratic elections, meanwhile tackling the security concerns too. Every application of Computer Science and this crucial issue is behind our motivation to develop VoteBlocks

2 System Requirements

- Hardware

    Processor: i5 or better

    RAM: 8 gigabytes or greater

- Software

    Operating System: Any

    Python: 3.6 or greater

3 Project Flow

The voting flow consists of two groups of processes that are dealt with in stages: electoral and counting. We put forward stringent yet enforceable guidelines to ensure fair elections.

3.1 Electoral Process

3.1.1 Voter Registration

All voters must be registered at three separate centres with the same information. The logistical challenges are recognised, but this is crucial to keeping the process fair and democratic. Each centre must follow strict guidelines in the registration of voters, and ask for three separate pieces of identification and proofs of address. After the preliminary registration process, a synchronisation procedure must be conducted to remove inconsistent data and recapture missing or incorrect data. Care must be taken by registration centres to not store or leak personal information, but rather store hashes to ensure individuality while protecting voter anonymity. This is achieved by hashing user details immediately on capture.

3.1.2 Polling Process

On election day, voters must report to any of the available designated polling centres (the decentralised nature also removes many restrictions currently in place today that enforce the casting of votes to a singular region). Voters must report with their names, their pre-designated password and unique voter-id. Each voter must cast a vote as early as possible (see tallying process). Once the designed block-size limit is achieved, centres must mine the casted votes to add them to the network. These unconfirmed votes will be broadcast on the blockchain to allow public mining for purposes of trust distribution and decentralisation.

3.2 Tallying Process

Votes are tallied from a first-in first-out principle. For each voter-hash stored, a process is conducted to ensure that there are no other votes cast with a later time-stamp. If such votes exist, they are discarded immediately. This is in accordance with the blockchain double-spending problem solution [3]. A check is then performed to ensure that the corresponding voter-hash exists in more than two registration centre databases. If these tests pass, the vote is tallied towards the total for a candidate. The raw data of the blockchain with the specific voter-hashes is then broadcasted to the public to ensure transparency.

4 Blockchain Structure

The system is broken down into two major components. The block, and the chain. Each chain is made up a number of blocks, all interlinked with each other. Each block is populated with the transactions, i.e., votes in our model. [4]

When votes are cast, they are added to a list of unconfirmed transactions. When the block is to be mined, it will be put through a SHA-256 function numerous times in order to match a certain hash pattern. Once the right hash is formed using the nonce, it is added to the block.

When combined, these entities form blockchains, which are hosted and coordinated by a number of Python Flask servers, to ensure redundancy. This ensures that the election process can continue with the support of a simple plug-and-play system, where nodes and servers can be swapped out while in operation to fix issues or improve performance. This is beneficial as it provides maximum redundancy while allowing industry-standard operational performance.

4.1 Structure of a block

Each block consists of the following elements essential to recording votes and identifying individuality of each voter. SHA- 256 hashing is used, where the data of the block is first input into a string, and then computed after being Unicode encoded.

- Index - To enumerate each block
- Transactions - The given transactions in each block (votes)
- Timestamp - The timestamp of each block's creation
- Previous Hash - The hash of the previous block to facilitate the blockchain
- Nonce - The number only used once. Used to supplement the rest of the data to generate a desired hash pattern

4.2 Structure of the chain

We discuss the structure of the chain as a combined product of its properties and functions as listed.

4.2.1 Difficulty property

The difficulty is a simple integer that determines how hard it is to mine a block. The higher this integer is set, the more difficult it is to mine. This is accomplished by using the number of leading zeros to the hash. A difficulty of 2 will ensure that there are two leading zeros for each accepted hash.

4.2.2 Genesis block property

The genesis block is the first mined block of that blockchain. Here, we mine it with a list of empty transactions and null (0) data, and we add it to the chain.

4.2.3 Last block property

A built-in function of the blockchain to retrieve the last added block. Useful in many use-cases to check timestamps and retrieve previous hashes.

4.2.4 Add block methodology

A simple function to add a block to the blockchain which requires only the block object and the proof of work. The block is verified by checking the proof of work to ensure its validity, and ensuring that the previous block hash of the blockchain, and the previous hash property of the submitted block match.

4.2.5 Proof of work function

A simple function that tries different nonce values to satisfy the hash pattern requirements set by our difficulty criteria.

4.2.6 Add new transaction function

Appends a transaction to the unconfirmed transaction chain.

4.2.7 Proof verification function

The proof is verified by using the hash supplied and the SHA-256 hash computed on the spot.

4.2.8 Chain validity check function

We use the valid proof check on each and every block in the chain to check if the chain as a whole is valid. It is this process that gives the strength to the blockchain, because it is relatively easy to verify the validity of the chain, compared to creating a fake valid chain from scratch.

4.2.9 Mining function

A new block is created with the given unconfirmed transactions and the proof of work is computed. The add block method is then used to add the new block to the blockchain, and the unconfirmed transactions / votes are removed.

4.3 Structure of the API

4.3.1 New transaction reporting

A post request route that allows you to send a candidate choice and a voter-hash, validates the requirement, and adds the cast vote into the list of unconfirmed transactions.

4.3.2 Chain retrieval

A get request route that returns the entire blockchain.

4.3.3 Mining

A get request route that mines the block, negotiates consensus and announces blocks to the entire network.

4.3.4 Registration of new nodes

A post request route to register new nodes to the network and add it to a peer list. This internally calls other functions to create chain from data dumps and register with other nodes.

4.3.5 Addition of new blocks

A post request route to add blocks mined by other entities to the chain. The data is first verified and then added.

4.3.6 Pending transaction retrieval

A get request route to retrieve all unmined votes.

5 Conclusions and further research scope

The zero-knowledge structure of the blockchain and the efficacy of the Nakamoto consensus algorithm will ensure anonymous and decentralized elections. However, it is crucial that all stakeholders recognize that the concept of such a fair and decentralized election is subject to a fair registration process by a recognized central agency earlier in the electoral life-cycle. Though there exist various methodologies to decentralize the registration process to ensure end-to-end distribution of control, it is the view of the author that these ideas will not be implementable in countries with massive populations and low technological advancement and accessibility. There is scope for further research regarding implementing an end-to-end decentralized electoral process in developing nations.

References

[1] Fund, John H. Stealing elections: How voter fraud threatens our democracy. Encounter Books, 2008.

[2] Shashidhar, S. VoteBlocks. GitHub. https://github.com/sumukshashidhar/voting-blockchain-application, 2021.

[3] Hoepman, Jaap-Henk. ”Distributed double spending prevention.” International Workshop on Security Protocols. Springer, Berlin, Heidelberg, 2007.

[4] 3Blue1Brown. “But How Does Bitcoin Actually Work?” YouTube, 7 July 2017, www.youtube.com/watch?v=bBC-nXj3Ng4.

Appendices

Appendix 1 - Code written

Appendix 2 - Screenshots

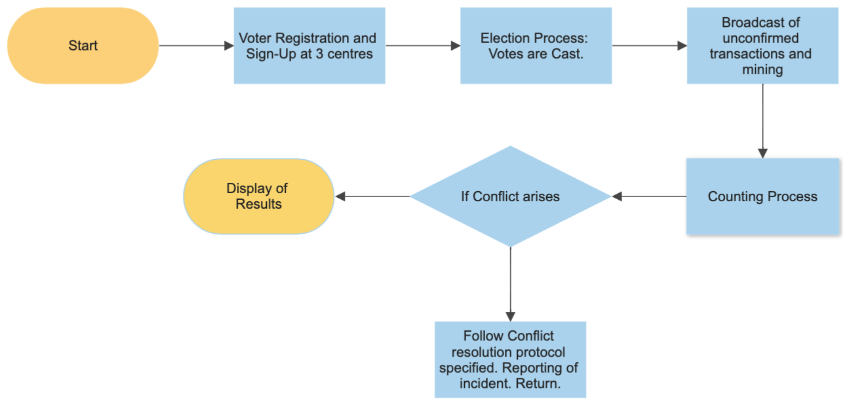
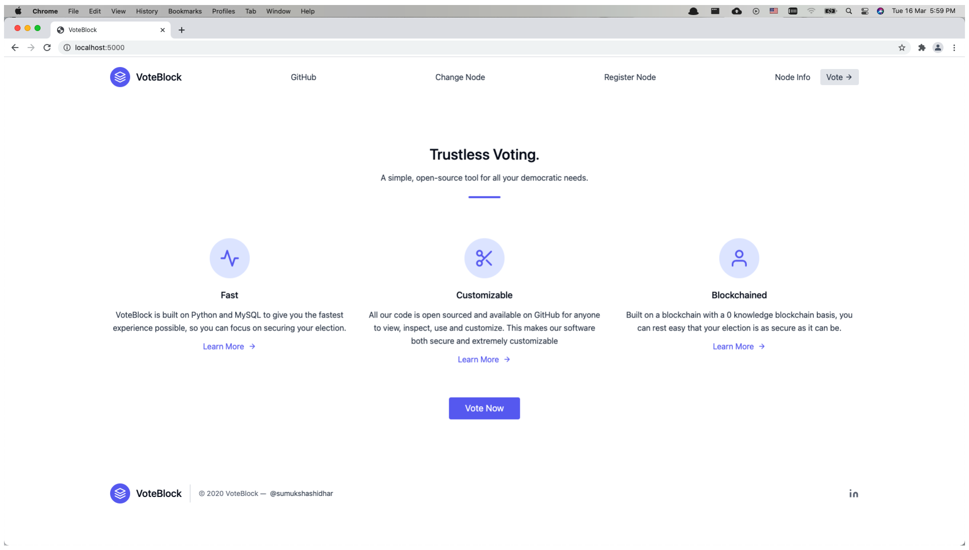

Start
Voter Registration and Sign-Up at 3 centres
Election Process: Votes are Cast.
Broadcast of unconfirmed transactions and mining
Counting Process
If Conflict arises
Display of Results
Follow Conflict resolution protocol specified. Reporting of incident. Return.

Figure 1: Flow of the entire registration process



Trustless Voting.

A simple, open-source tool for all your democratic needs.

Fast

VoteBlock is built on Python and MySQL to give you the fastest experience possible, so you can focus on securing your election.

Learn More →

Customizable

All our code is open sourced and available on GitHub for anyone to view, inspect, use and customize. This makes our software both secure and extremely customizable

Learn More →

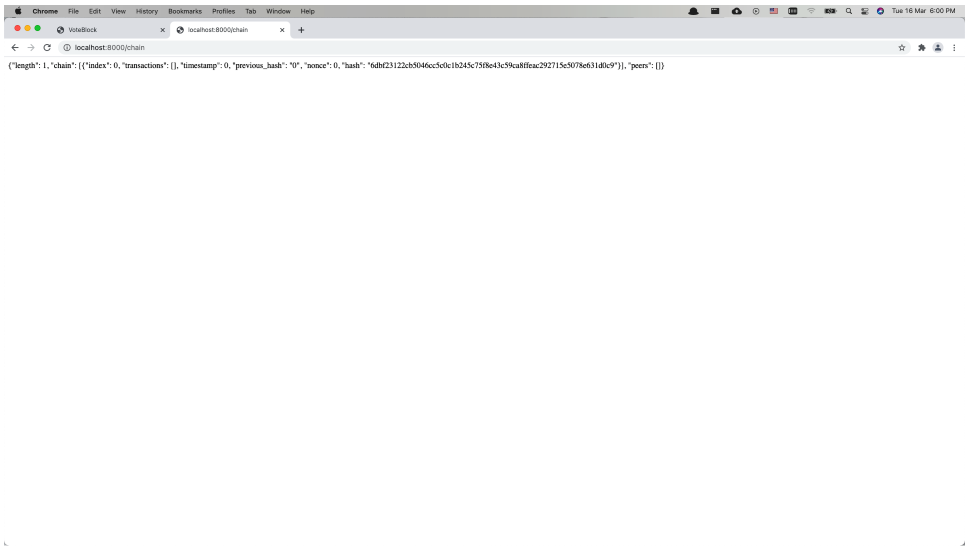
Blockchained

Built on a blockchain with a 0 knowledge blockchain basis, you can rest easy that your election is as secure as it can be.

Learn More →

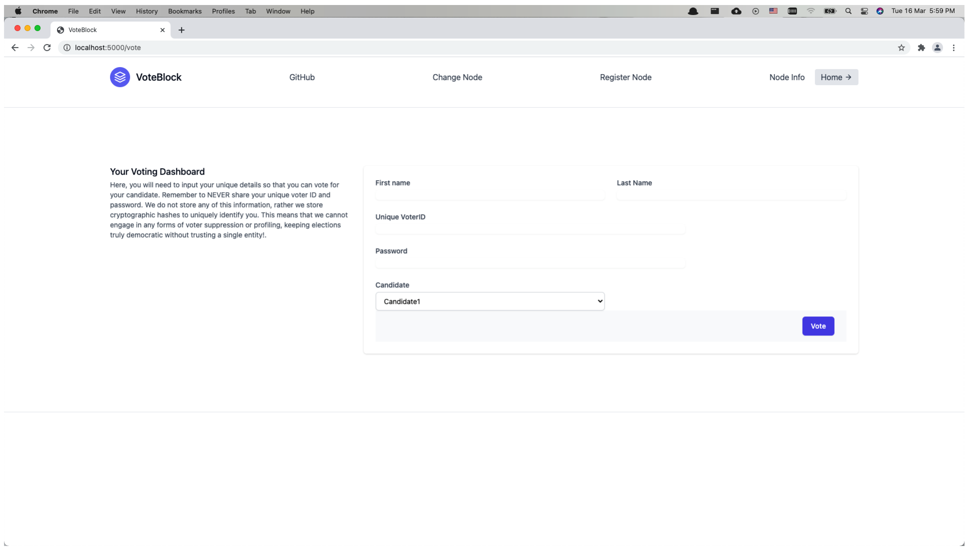
© 2020 VoteBlock — @sumukshashidhar

Figure 2: Home page of the voting application



{"length": 1, "chain": [{"index": 0, "transactions": [], "timestamp": 0, "previous_hash": "0", "nonce": 0, "hash": "6dbf23122cb5046cc5c0c1b245c75f8e43c59ca8ffeac292715e5078e631d0c9"}], "peers": []}

Figure 3: State of the blockchain initially

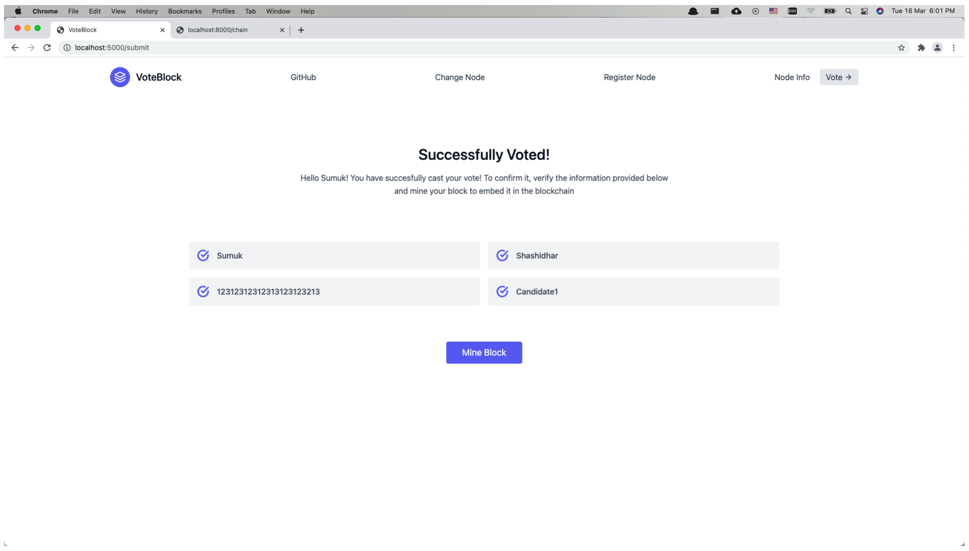


Your Voting Dashboard

Here, you will need to input your unique details so that you can vote for your candidate. Remember to NEVER share your unique voter ID and password. We do not store any of this information, rather we store cryptographic hashes to uniquely identify you. This means that we cannot engage in any forms of voter suppression or profiling, keeping elections truly democratic without trusting a single entity!.

First name
Last Name
Unique VoterID
Password
Candidate
Candidate1

Figure 4: The preliminary voting screen for the voter

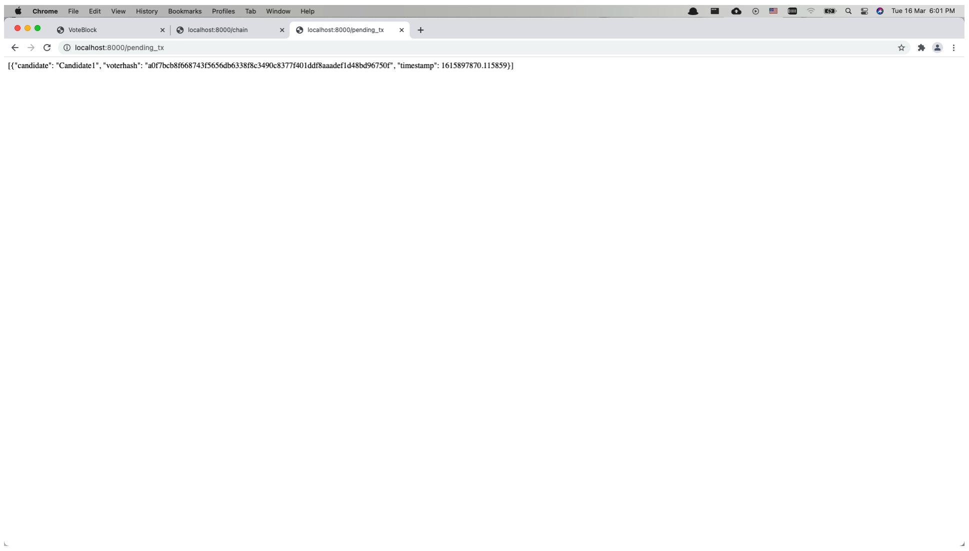


Successfully Voted!

Hello Sumuk! You have succesfully cast your vote! To confirm it, verify the information provided below and mine your block to embed it in the blockchain

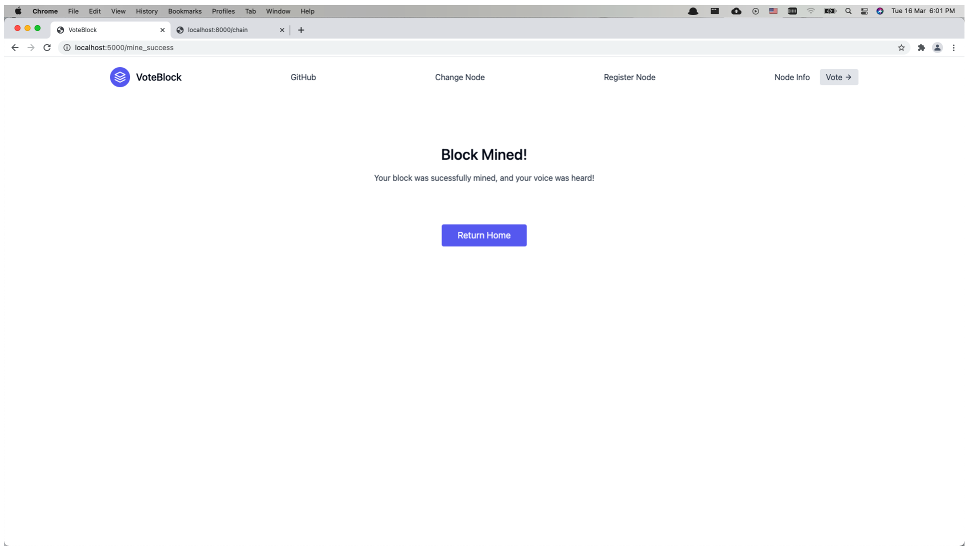
✓ Sumuk	✓ Shashidhar
✓ 12312312312313123123213	✓ Candidate1

Figure 5: Screen after a succesful vote



```
[{"candidate": "Candidate1", "voterhash": "a0f7bcb8f668743f5656db6338f8c3490c8377f401ddf8aaadef1d48bd96750f", "timestamp": 1615897870.115859}]
```

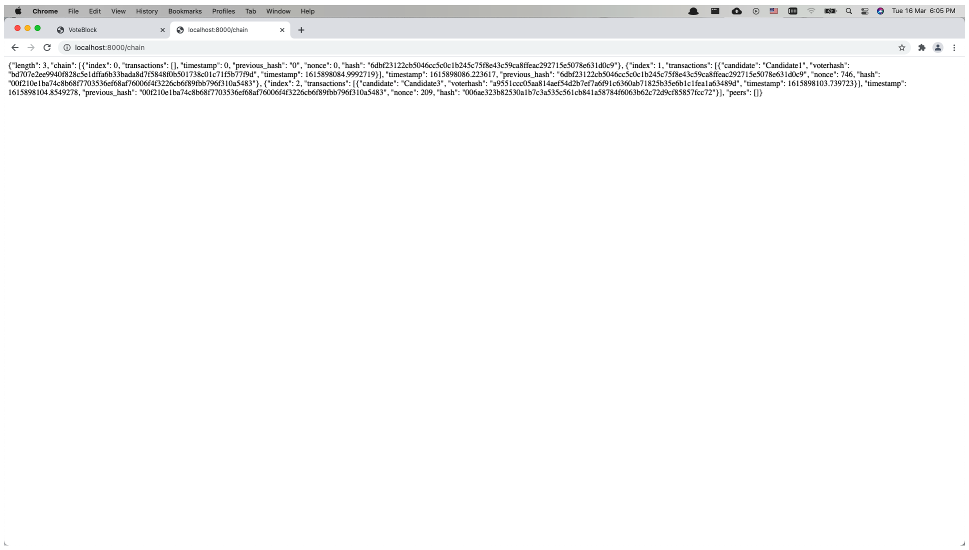
Figure 6: State of the pending transactions before mining



Block Mined!

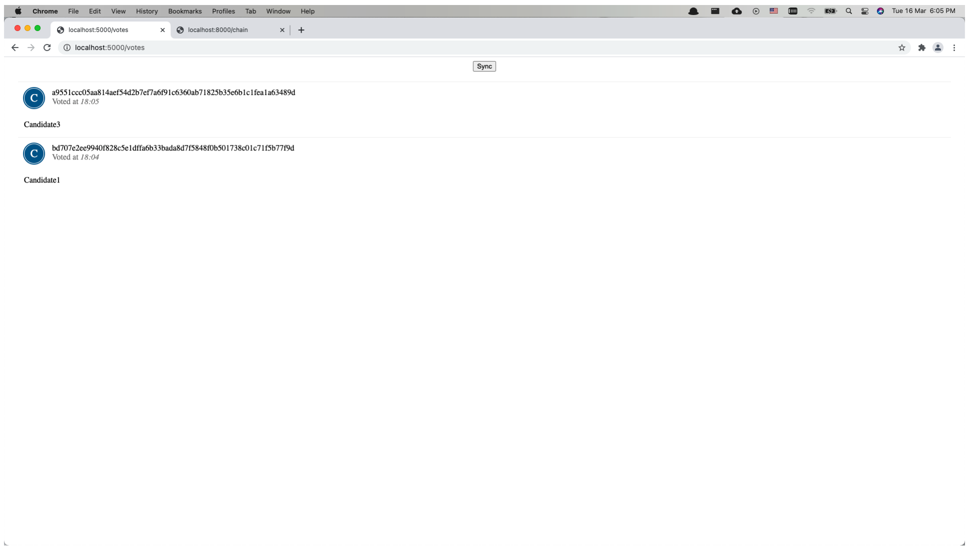
Your block was sucessfully mined, and your voice was heard!

Figure 7: Screen after mining



```
{"length": 3, "chain": [{"index": 0, "transactions": [], "timestamp": 0, "previous_hash": "0", "nonce": 0, "hash": "6dbf23122cb5046cc5c0c1b245c75f8e43c59ca8ffeac292715e5078e631d0c9"}, {"index": 1, "transactions": [{"candidate": "Candidate1", "voterhash":
"bd707e2ee9940f828c5e1dffa6b33bada8d7f5848f0b501738c01c71f5b77f9d", "timestamp": 1615898084.9992719}], "timestamp": 1615898086.223617, "previous_hash": "6dbf23122cb5046cc5c0c1b245c75f8e43c59ca8ffeac292715e5078e631d0c9", "nonce": 746, "hash":
"00f210e1ba74c8b68f7703536ef68af76006f4f3226cb6f89fbb796f310a5483"}, {"index": 2, "transactions": [{"candidate": "Candidate3", "voterhash": "a9551ccc05aa814aef54d2b7ef7a6f91c6360ab71825b35e6b1c1fea1a63489d", "timestamp": 1615898103.739723}], "timestamp":
1615898104.8549278, "previous_hash": "00f210e1ba74c8b68f7703536ef68af76006f4f3226cb6f89fbb796f310a5483", "nonce": 209, "hash": "006ae323b82530a1b7c3a535c561cb841a58784f6063b62c72d9cf85857fcc72"}], "peers": []}
```

Figure 8: Blockchain dump after voting



a9551ccc05aa814aef54d2b7ef7a6f91c6360ab71825b35e6b1c1fea1a63489d
Voted at *18:05*

Candidate3

bd707e2ee9940f828c5e1dffa6b33bada8d7f5848f0b501738c01c71f5b77f9d
Voted at *18:04*

Candidate1

Figure 9: List of all votes and voters with timestamps

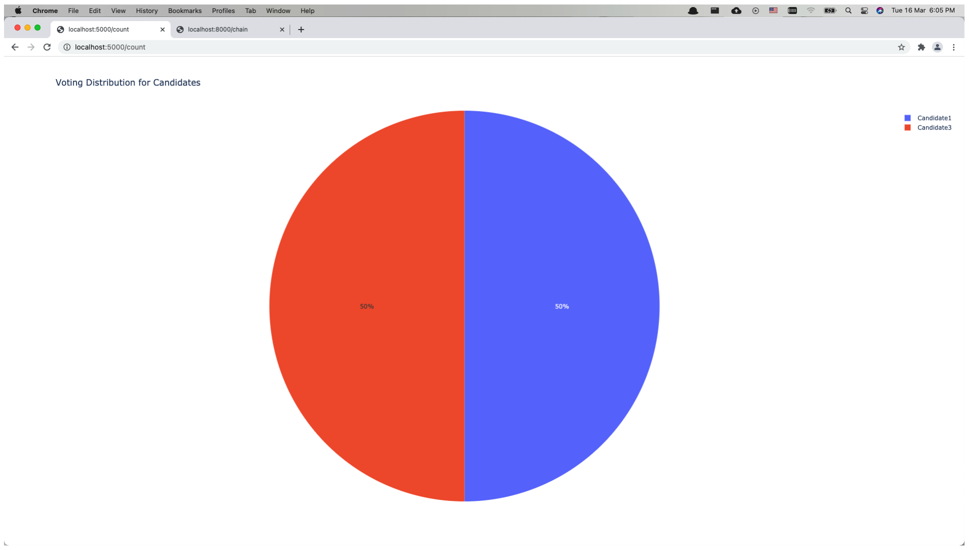
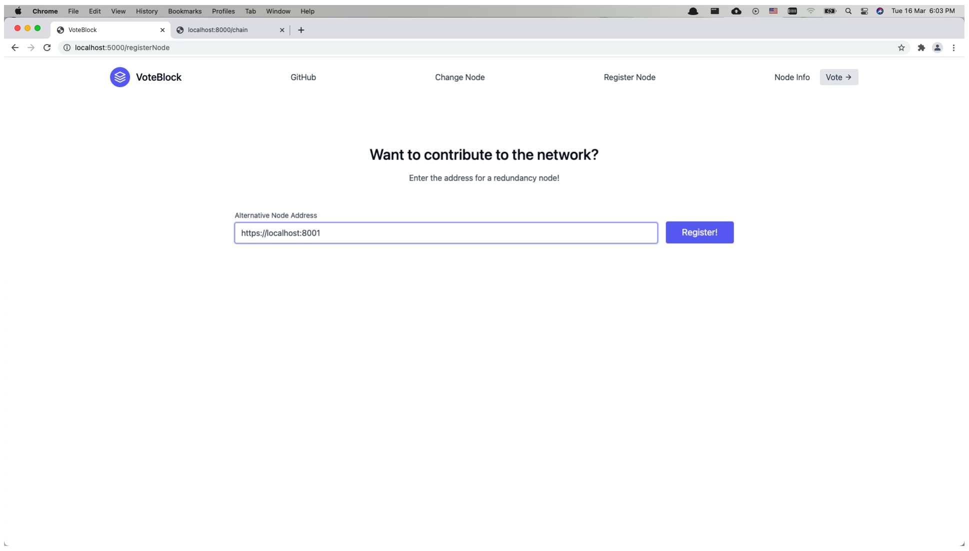


Figure 10: Graphical display of the votes

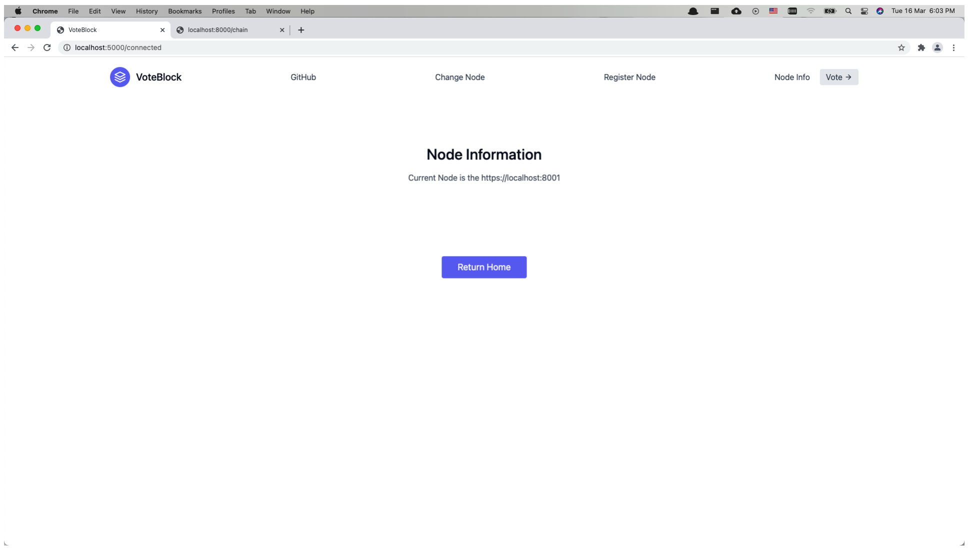


Want to contribute to the network?

Enter the address for a redundancy node!

Alternative Node Address: https://localhost:8001

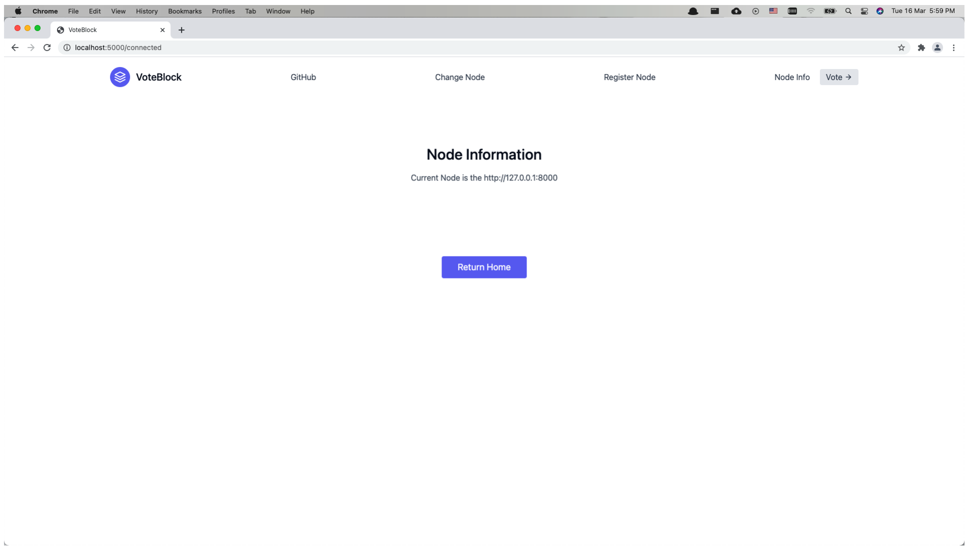
Figure 11: Alternate node registration screen (redundancy purposes)



Node Information

Current Node is the https://localhost:8001

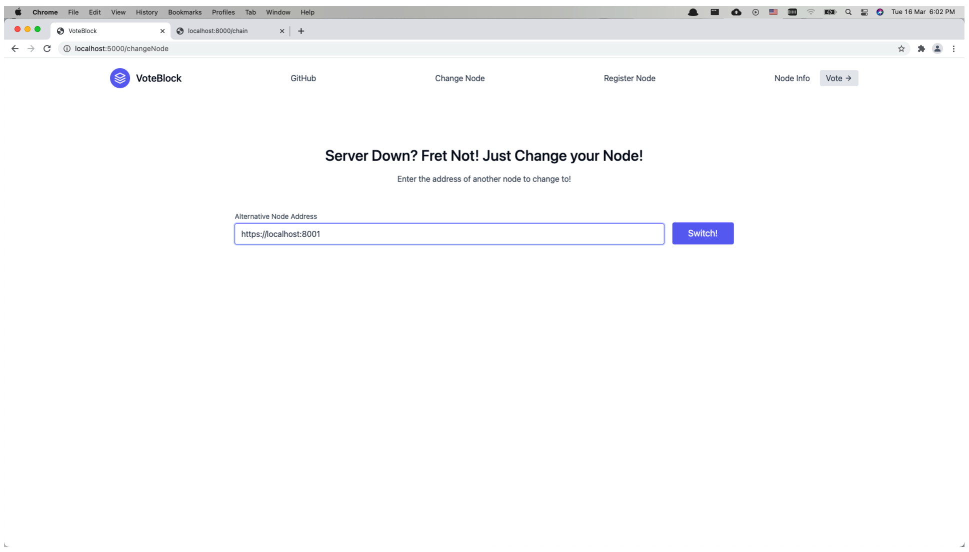
Figure 12: Confirmation of node registration



Node Information

Current Node is the http://127.0.0.1:8000

Figure 13: Initial State before switching nodes



VoteBlock GitHub Change Node Register Node Node Info Vote →

Server Down? Fret Not! Just Change your Node!

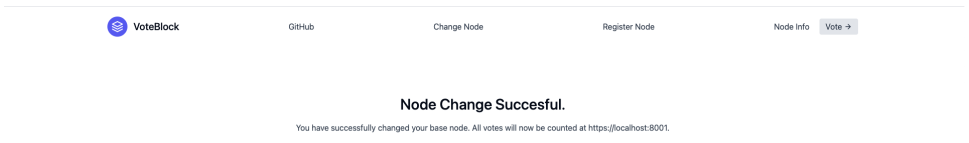
Enter the address of another node to change to!

Alternative Node Address

https://localhost:8001

Switch!

Figure 14: Server switching screen



VoteBlock GitHub Change Node Register Node Node Info Vote →

Node Change Succesful.

You have successfully changed your base node. All votes will now be counted at https://localhost:8001.

Figure 15: Successful Server change screen