True Democratic Consensus Mechanism for DAO Governance

Completed by Group 1

Abeeha Aleem, Sangamithra Bala Amuthan, Jiale Shang, Mazen Wael & Abdullah Mohamed

Link to Deployed System: https://quadradao-app.azurewebsites.net

1. Title of Project: True Democratic Consensus Mechanism for DAO Governance
2. Introduction
   i. Background: Voting and Democracy have existed since their development in 508B.C. Ancient Greece to give the power of governance to the hands of the masses. Voting allows individual users to voice their opinion on current topics and how they should be executed. With the principles of voting in mind, many nations have a democratic government formed upon the votes cast by all citizens. Due to deviation from democracy and poor voting practices at large scales, voters are choosing not to vote in elections. The decline is constant throughout the years. In similar pursuits, each blockchain decentralized autonomous organization (DAO) uses a voting process so that members of the DAO can vote on governance and relevant proposals. The benefits of DAO voting are similar as they allow members to truly be in charge of the decisions their organization makes. Despite this sense of ownership, blockchains DAO voting can also decline due to unfair practices.
   ii. Motivation: Deploy a consensus mechanism to optimize the voter experience, trust in elections and turnout. The goal is to return true democracy to voting by instilling fairness for all voters.
   iii. Context: The DAO uses a consensus system to make any decision for the organization. This system consists of a proposal and the opportunity for each member to vote on their preferred outcome. The outcome implemented by the DAO is determined by most votes at the end of the consensus. Although the goal is to best represent all the members of the DAO, voting mechanisms often favour the wealthy since they are able to purchase more voting tokens which can imbalance the majority and implementation of the proposal overall. Not to mention, electronic voting mechanisms are also susceptible to external security attacks, hackers and corruption. This lack of itegrity and fairness in the DAO vote deters voter engagement and further misrepresents the outcome of the vote.
3. Problem Statement
   i. We propose a blockchain DAO based voting system with the integration of a time decay function within a quadratic voting system. This innovative approach, enhanced by a whitelist to verify eligible voters and mechanisms for anonymous voting to protect voter privacy, aims to boost early voter participation, increase overall turnout, and create a more secure, inclusive electronic electoral process.

This combination is anticipated to better reflect the community's preferences and increase trust in the consensus system. The system incorporates time decay and anonymous voting.

4. Solutions
   i. Key Features

   Proposal: The consensus system allows users to submit a proposal to be voted on. Alongside the proposal, the voting options can also be submitted so that votes are cast towards a specific option. The proposal also has an expiry time so that voting opens and closes appropriately. This also prompts users to cast their votes within the timeframe to reach a decision. The submitter can also put a description of the proposal and provide any additional information needed. This mimics typical voting in real life such as for the election day of governmental elections. The proposal system can be beneficial to allow clear decisions to be made in an efficient manner representative of the members' opinion without the need to hours of boardroom discussions.

   Voting: The system allows users to log in and view all proposals and cast their votes for each proposal. The users can vote on the proposals by selecting any of the options provided for the proposal. The votes must be cast within the timeframe before the proposal expires. The voting is anonymous so members cannot see the specific option selected by other, only the total number of votes for each option. The voters are expected to vote using time delay and anonymous voting.

   Accounts: the consensus system requires the registration of all users with a valid MetaMask wallet. This extra privacy ensures only concerned users can access the system. The need for a MetaMask wallet also adds a layer of protection from bots who may spam the system. The need to register for the account also can help users keep track of current proposals and previous votes all in one place. The account for each user is also part of the user experience and adds to the feeling of membership and ownership which could motivate the user to engage in voting.

   Dark & Light Mode: The system features both light and dark modes as part of the user interface. This can be ergonomic as it promotes efficiency and comfort for users. This is also an accessible solution as some users require a certain lighting condition due to visual abilities and impairments. The dark and light mode settings enhance the user experience of the system promoting user engagement which will carry over to the voter engagement.

   ii. The Demo Video is linked in the GitHub Repo
   iii. Account Types

General User: General users also knows as voters of the system are able to login to the system. View all proposals that are submitted, current and historical. General voters can view all the votes for current and historical proposals. They can vote on any current proposals. General voters can request NFTs.

Admin Accounts: Admin users can do all the functions as general users and have additional capabilities. Admis users can submit proposals for all users to vote on. Admins are able to grant NFTs to general users. Admin accounts can change the status of any general user account to an admin account.

iv. Analytics Dashboard: The analytics of numbers of votes can be seen in each proposal. This is evident in the demo video in section 2.

v. Due to the nature of the system deployed the following values are improved on compared to traditional voting mechanisms.

Availability: Using Blockchain voting allows technology and voting from anywhere access to a device and stable internet connection.

Voting Goals: Voting goals and targets can be monitored easily by members of the DAO governance. The easy to use system can increase voter turnout to meet voting goals.

Transparency: The system allows the voting process to be transparent by clearly displaying all open and closed proposals and the number of votes per option at any given point in time.

Equipment: Use of this blockchain voting system allows higher user voting with limited resources and personnel needed.

Responsibility: The robust system increases responsibility for voting in a timely yet informed manner.

Trust: The system with this improved voting mechanism and limited chance for internal corruption improves voter trust in the governance.

Integrity: The system increases integrity by verification of users and removing the chance of attacks

5. Instructions to Use

i. Local setup

The following instructions include the installation, configuration and command line instructions for a local setup of the application.

Installation

install node

install Truffle and Ganache globally npm install -g truffle ganache-cli

install openzeppelin: npm install @openzeppelin/contracts

install Metamask

Configeration

In /truffle folder, add .env file, add PUBLIC_KEY=<Your_MetaMask_PUBLICKEY> and PRIVATE_KEY=<Your_MetaMask_PRIVATEKEY>, if you are running on a local ganaceh-cli, put one of the Address generated by ganache in PUBLIC_KEY, this will be your admin address.

Run Dev Server

In command line, run ganache-cli, or launch ganache desktop.

Open Metamask, add the ganache network (here HTTP://127.0.0.1:8545) and import the first private key generated by ganache.

On a seperate terminal window, run cd truffle then truffle migrate --network development --reset

In root folder, npm run start

ii. Required Libraries

The required libraries are truffle, ganache and openzepplin. Tailwind CSS with the appropriate azure account is also required. The installation commands and links are included below. MetaMask wallet web extension is also required.

Library Installation Commands

install Truffle and Ganache globally npm install -g truffle ganache-cli

install openzeppelin: npm install @openzeppelin/contracts

MetaMask Wallet Instructions (Cannot be done from CLI)

https://support.metamask.io/hc/en-us/articles/360015489531-Getting-started-with-MetaMask

Tailwind CSS: https://tailwindcss.com/ To deploy the Dapp on Azure Portal, you need a workable Azure(https://portal.azure.com/) account.

iii. How to Run

The system can be used with the following link:

https://quadradao-app.azurewebsites.net. The features available for use by for each user type are described in the section below.

iv. User Types

General User:

- Login to Access the System
- View all Current and Historical Proposals
- View All votes on current and historical proposals
- Vote on Proposals
- Request NFT Access

Admin Accounts:

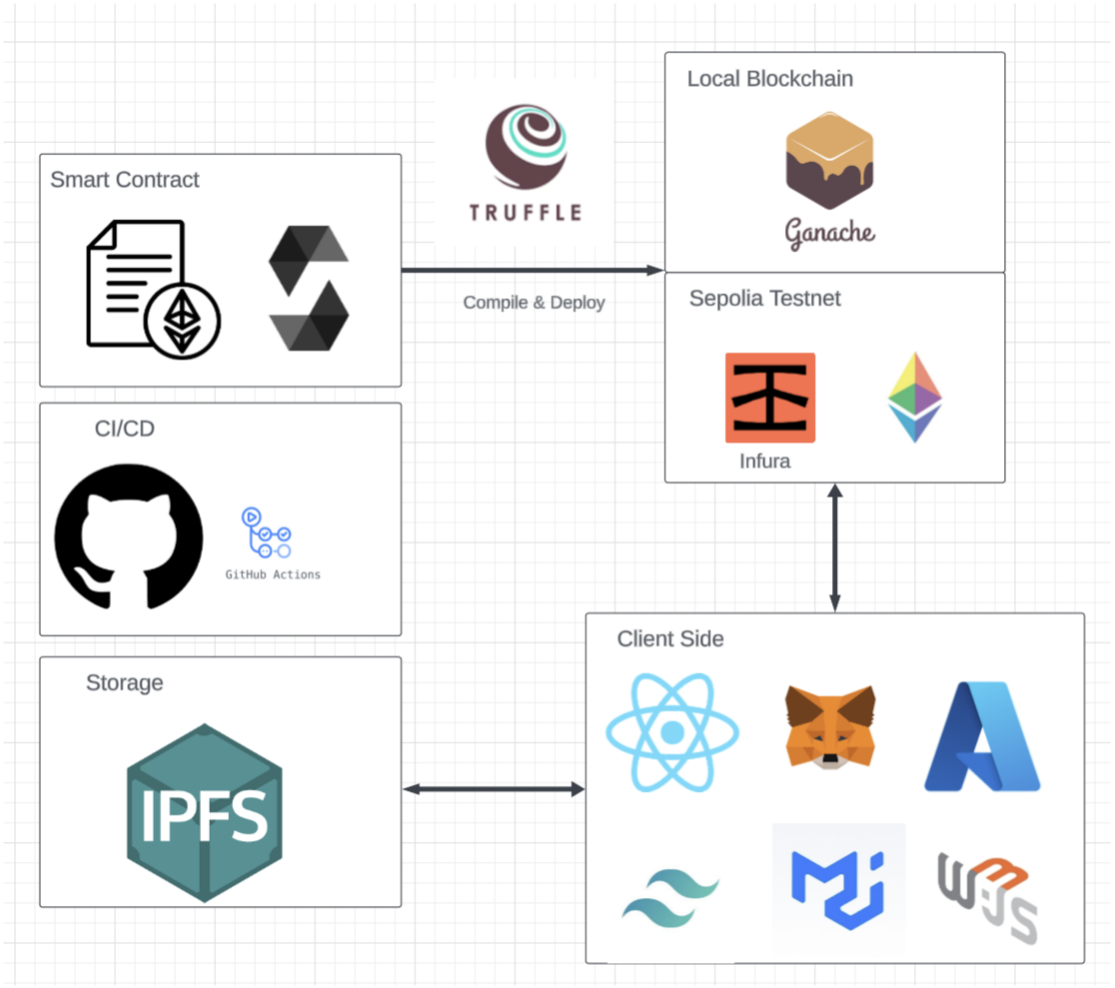
- All functions as general users
- Submit proposals for all users to vote on
- Grant NFTs to general users.
- Change the status of any general user account to an admin account

v. Metamask: This system uses MetaMask and needs to be linked with Sepolia Testnet in production, and ganache network (here HTTP://127.0.0.1:8545) in development.

6. Contribution

i. Architecture

The architecture diagram is below. The architecture comprises of several functional components. The initial smart contracts written in Solidity are bridged to the local blockchain using Truffle. The local blockhain for development is Ganache Based. The production blockchain network used is Sepolia Testnet. The client side of the application uses react to build the front end and MetaMask to authorize voters. The front end also requires Web3js, tailwindcss and MaterialUI in order to provide a seamless front end experience. GitHub was used for CI/CD due to ease of collaboration and development. Finally, an IPFS storage was used for storing data such as user accounts. The use of GitHub allows any other developers to use the system and work upon it by submitting a pull request.



ii. Local setup instructions

The required libraries are truffle, ganache and openzepplin. The installation commands and links are included below. MetaMask wallet web extension is also required.

Library Installation Commands

install Truffle and Ganache globally npm install -g truffle ganache-cli

install openzeppelin: npm install @openzeppelin/contracts

MetaMask Wallet Instructions (Cannot be done from CLI)

https://support.metamask.io/hc/en-us/articles/360015489531-Getting-started-with-MetaMask

Testing

Any level of user testing can be completed on the prod version of the system using the following link: https://quadradao-app.azurewebsites.net

New Smart Contact

In order to deploy a new smart contract run the following truffle command. This command uses truffle to convert the solidity based smart contract into a JSON like format called “abis”.

```
Truffle migrate -- reset
```

7. Licensing
    i. This system is developed using The MIT license. This License allows others to use the software for educational purposes without any warranty or owness on the developers.
8. Credit & Acknowledgements
    i. This system is built upon the DAO voting system created by Darlington “Daltonic” Gospel in “dappVote” repository and associate tutorial. The developers of this system would like to thank Daltonic for allowing us to leverage the voting system to create proposals and vote.
    ii. The developed sytem used the components for DAO voting from “dappVote” and constructed a more robust user interface with different themes. The developed system also integrated more robust voting mechanisms such as anonymous voting and whitelisting to create more robust voting measures and work towards the goal of increased transparency and voter engagement.