Polling System

January 24, 2024

Author: Martin Velasquez

1. Introduction: Revolutionizing Polling Through Blockchain

In the realm of technological innovation, blockchain has emerged as a transformative force, reshaping conventional systems with its decentralized and transparent architecture. My project delves into the application of blockchain technology to create a decentralized polling system, addressing inherent challenges in traditional methods. Deployed on the Ethereum blockchain, my Polling System smart contract ensures the integrity and transparency of the voting process. Users interact with the contract through Remix, connecting their wallets to cast secure and verifiable votes. This initiative exemplifies the potential of blockchain to redefine trust and accountability in processes like polling.

2. Technology and Tools

2.1 Smart Contract Development with Solidity

My project hinges on the Solidity programming language, specifically designed for creating smart contracts on the Ethereum blockchain. Solidity enables me to define the logic of the Polling System contract, incorporating features like options for voting, voter registration, and secure tallying of votes. The language's syntax and structure are tailored for the blockchain environment, ensuring the reliability and security of our decentralized application.

2.2 Remix Development Environment

Remix serves as my primary development environment, providing a web-based interface for creating, testing, and deploying smart contracts. With its intuitive design, Remix simplifies the development process, allowing me to write, compile, and deploy Solidity code seamlessly. Its integrated debugger and testing tools facilitate efficient troubleshooting and ensure the robustness of our smart contract before deployment.

2.3 Ethereum Blockchain Network

My choice of the Ethereum blockchain as the underlying infrastructure for the voting system is motivated by its widespread adoption and robust smart contract capabilities. Ethereum's decentralized nature and support for smart contracts make it an ideal platform for implementing transparent and tamper-resistant voting processes. We interact with the Ethereum blockchain on a test network like Ganache to deploy and execute our smart contract.

2.4 Ganache: A Local Blockchain for Development and Testing

Ganache is a local blockchain emulator designed for Ethereum development and testing purposes. It allows developers to simulate the behavior of a real blockchain network in a controlled environment, providing a set of predefined accounts with associated private keys and a configurable number of Ether for testing. Ganache is invaluable during the development phase, enabling developers to deploy, interact with, and test smart contracts without incurring the costs and time associated with deploying on the Ethereum mainnet. Its user-friendly interface and integration with popular development tools make it a go-to choose for Ethereum developers.

2.5 MetaMask

MetaMask is a browser extension that acts as an Ethereum wallet, allowing users to manage their Ether and interact with decentralized applications (DApps) directly from their web browsers. It provides a convenient bridge between web applications and the Ethereum blockchain. By seamlessly connecting to Remix or any Ethereum-enabled platform, MetaMask enables users to deploy smart contracts, vote, and engage with decentralized applications effortlessly.

3. Smart Contract Design

The smart contract is structured to include two main data structures: Option and Voter. The Option struct represents each voting choice, containing the option's name and the count of votes it has received. The Voter structure ensures that each address can vote only once by tracking whether the voter has already cast a vote.

3.1 Constructor

The constructor initializes the options for voting during the deployment of the contract. In my example, "Iyad Koteich" and "Justin Trudeau" are the predefined choices.

```
constructor() {

        options.push(Option({ name: "Iyad Koteich", voteCount: 0 }));
        options.push(Option({ name: "Justin Trudeau", voteCount: 0 }));
    }
```

3.2 Vote Function

The vote function allows a voter to cast a vote for a specific option. It checks that the voter has not voted before and increments the vote count for the chosen option.

```
function vote(uint256 _optionIndex) external hasNotVoted validOption(_optionIndex) {

    voters[msg.sender].hasVoted = true;

    options[_optionIndex].voteCount++;

    emit Voted(msg.sender, _optionIndex);

}
```

3.3 Modifier for Validation

Two modifiers, hasNotVoted and validOption, enhance the security and validity of the contract. hasNotVoted ensures that a voter can only cast one vote, and validOption verifies that the chosen option index is within the valid range.

```
modifier hasNotVoted() {

    require(!voters[msg.sender].hasVoted, "You have already voted");

    _;

}


modifier validOption(uint256 _optionIndex) {

    require(_optionIndex < options.length, "Invalid option");

    _;

}
```

3.4 Getter Functions

Additional functions, such as getVoteCount and getOptionCount, provide a way to query the state of the contract. These functions enable users to retrieve the count of votes for a specific option and the total number of available options, respectively.

```
function getVoteCount(uint256 _optionIndex) external view validOption(_optionIndex)
returns (uint256) {
```

```
    return options[_optionIndex].voteCount;
}

function getOptionCount() external view returns (uint256) {
    return options.length;
}
```

3.5 Events

The contract emits the Voted event each time a vote is cast, providing transparency, and allowing external applications to listen for and react to voting activities.

```
event Voted(address indexed voter, uint256 indexed optionIndex);
```

4. Contract Deployment and Configuration

As the developer, I initiated the deployment of the "PollingSystem" smart contract using the Remix IDE, Ganache, and MetaMask. In Remix, I carefully configured the compiler version and environment settings, opting for "Injected Web3" to seamlessly connect with MetaMask. After confirming that MetaMask was linked to the same Ethereum network as Remix, I deployed the contract, with MetaMask prompting me to approve and confirm the deployment transaction. Observing the deployment process on Remix and the MetaMask interface, I recorded the contract address, a critical identifier for future interactions.

To test the contract's functionality, I interacted with it through Remix, casting votes and querying results using the provided functions. The configuration ensured a smooth connection between Remix, MetaMask, and the Ethereum network, offering a user-friendly and transparent experience. Debugging tools in Remix proved valuable during the development phase, allowing me to identify and address any issues that arose. Throughout this process, the decentralized nature of the smart contract guaranteed tamper-resistant and auditable voting results, fostering trust in the integrity of the polling system. This end-to-end journey highlighted the effectiveness of blockchain technology in creating a reliable and transparent decentralized polling solution.

5. Interaction with the Contract:

Once the "PollingSystem" smart contract is deployed, users can seamlessly engage with the decentralized polling system using MetaMask and Remix. As an end user, the process begins by connecting my MetaMask wallet to Remix, ensuring that my Ethereum address is linked to the deployed contract. With MetaMask authenticated, I navigate to Remix's user interface, where I'm presented with the options to cast my vote.

To participate in the poll, I select the preferred voting option and initiate the voting transaction through Remix. MetaMask promptly prompts me to confirm the transaction, and after approval, the vote is securely recorded on the Ethereum blockchain. The decentralized nature of the smart contract ensures that each user can only vote once, fostering trust in the integrity of the polling system.

Using Remix's querying functions, I can also retrieve real-time information about the poll's status, checking the vote count for each option and the total number of available choices. The transparent and tamper-resistant design of the smart contract guarantees an auditable and fair voting process. This seamless interaction between MetaMask and Remix demonstrates the user-friendly and secure experience offered by decentralized applications powered by blockchain technology.

6. Tests and Results

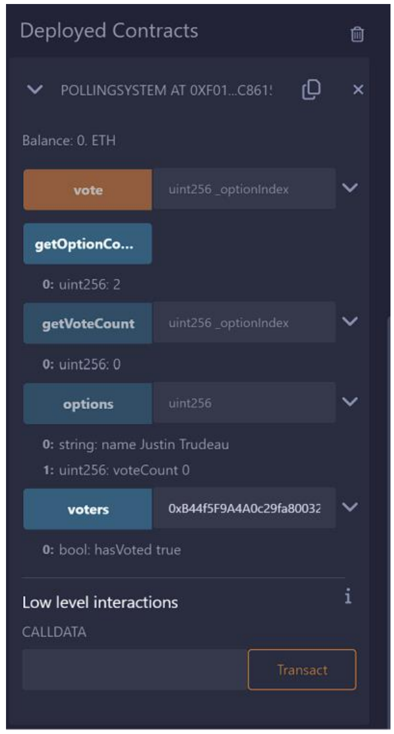
The testing phase yields valuable insights into the contract's resilience and adherence to specified rules. Any identified issues are addressed through iterative development and debugging. Successful tests confirm the contract's ability to securely record and tally votes, providing users with a trustworthy and transparent polling experience. The transparent nature of the blockchain ensures that results are verifiable and immune to manipulation, establishing confidence in the decentralized polling system's integrity.

6.1 The code

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract PollingSystem {

    struct Option {
        string name;
        uint256 voteCount;
    }


    struct Voter {
        bool hasVoted;
    }
```

```
    mapping(address => Voter) public voters;

    Option[] public options;

    event Voted(address indexed voter, uint256 indexed optionIndex);

    modifier hasNotVoted() {
        require(!voters[msg.sender].hasVoted, "You have already voted");
        _;
    }

    modifier validOption(uint256 _optionIndex) {
        require(_optionIndex < options.length, "Invalid option");
        _;
    }

    constructor() {

        options.push(Option({ name: "Iyad Koteich", voteCount: 0 }));
        options.push(Option({ name: "Justin Trudeau", voteCount: 0 }));
    }

    function vote(uint256 _optionIndex) external hasNotVoted validOption(_optionIndex)
{
        voters[msg.sender].hasVoted = true;
        options[_optionIndex].voteCount++;
        emit Voted(msg.sender, _optionIndex);
    }

    function       getVoteCount(uint256       _optionIndex)       external       view
validOption(_optionIndex) returns (uint256) {
        return options[_optionIndex].voteCount;
    }

    function getOptionCount() external view returns (uint256) {
        return options.length;
    }
}
```

6.2 Deployed Contract

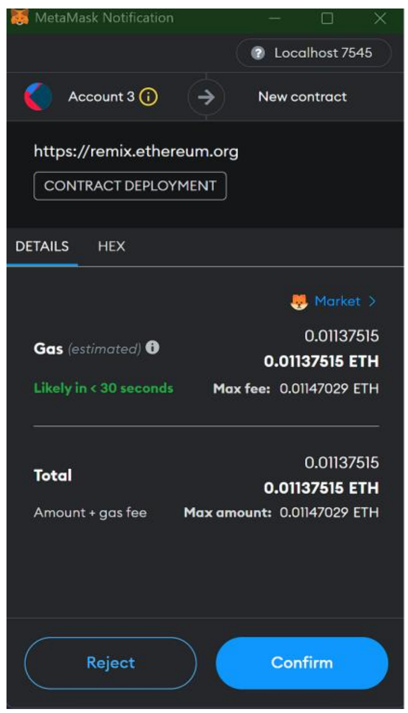


6.2.1 Option Function

6.2.2 Vote Function

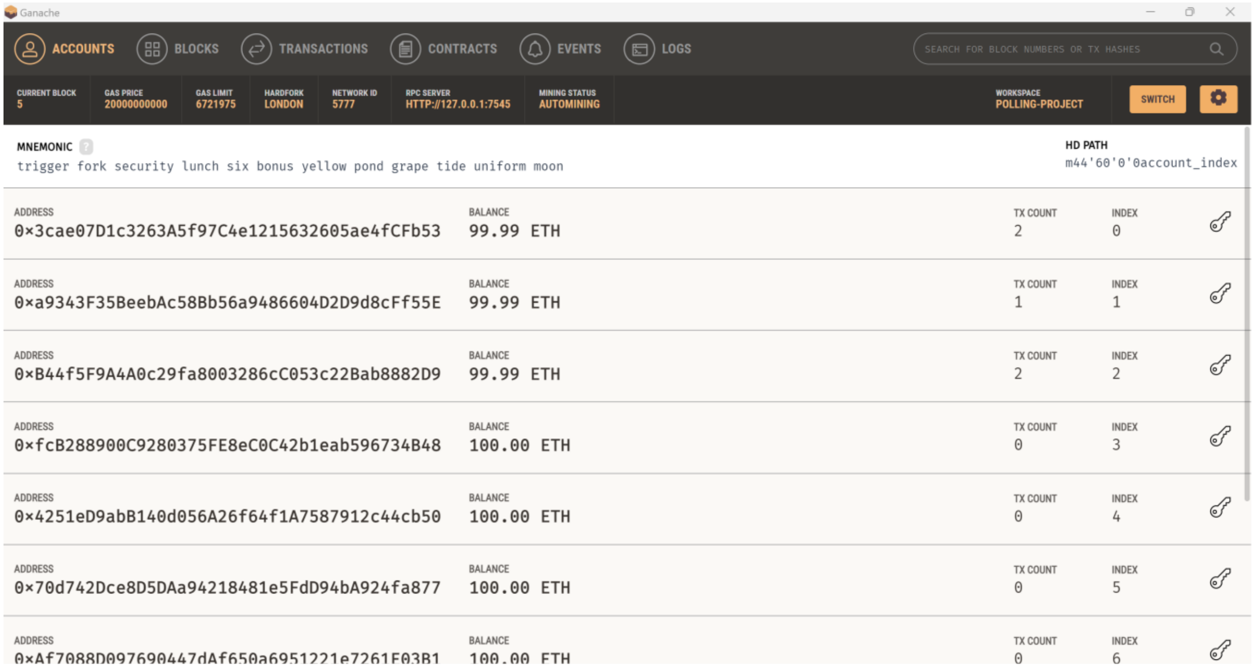
6.2.3 GetVoteCount Function

6.3 Metamask

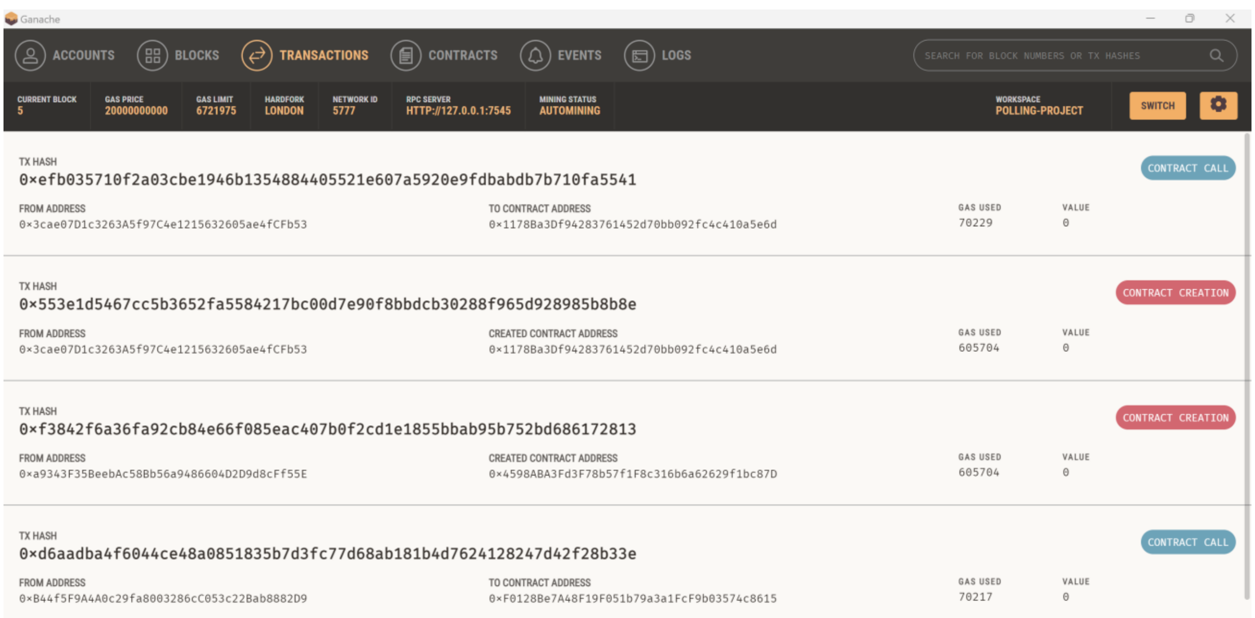


6.4 Ganesh

6.4.1 Accounts



6.4.2 Transactions



7. Conclusions

The decentralized polling system demonstrates the transformative potential of blockchain technology in ensuring transparency, security, and trust in voting processes. The tamper-resistant nature of the smart contract, validated through rigorous testing, solidifies its role as a robust and reliable platform for decentralized polls.

Looking forward, the Polling System serves as a testament to the broader applicability of blockchain in democratic processes. By leveraging the decentralized and transparent features of blockchain technology, this project contributes to the ongoing discourse on enhancing the integrity of voting systems. As blockchain continues to evolve, the lessons learned from this project pave the way for further innovations in decentralized governance and trust-building mechanisms.

8. Bibliography

- Ganache. (n.d.). Retrieved from https://www.trufflesuite.com/ganache
- MetaMask. (n.d.). Retrieved from https://metamask.io/
- Remix IDE. (n.d.). Retrieved from https://remix.ethereum.org/
- National Institute of Standards and Technology. (2012). NIST Special Publication 800-30: Guide for Conducting Risk Assessments. Retrieved from https://csrc.nist.gov/publications/detail/sp/800-30/rev-1/final
- ISO/IEC 27005:2018. (2018). Information technology - Security techniques - Information security risk management. Retrieved from https://www.iso.org/standard/73236.html
- Buterin, V. (2013). Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. Retrieved from https://ethereum.org/whitepaper/
- Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction. Princeton University Press.