



Diego Lauricella

# Start2Impact – Ethereum Web3 Project "Voting Dapp"

Deployed at: <https://votingdapps2i.netlify.app/>

GitHub repo: <https://github.com/diegoddie/VotingDapp-Start2ImpactProject>

Smart contract address: 0x3E29E6A88b20A4a0b96c4fbDF524A338576caA98



# Overview

The Voting DApp allows users to participate in the voting process for electing the next mayor of the city. The application is built on the Ethereum blockchain (SEPOLITA Testnet), ensuring decentralization and transparency in the voting process.

Users have 90 days to connect their Metamask wallet to the DApp and cast their vote for their preferred candidate. The blockchain technology ensures the security and immutability of the voting records, making it resistant to manipulation and fraud.



**Diego Lauricella**

# Tech Stack



- Hardhat
- Solidity
- React
- Ethers.js

- Thirdweb
- Tailwindcss
- DaisyUI



**Diego Lauricella**

# Key Features



- **Countdown:** A countdown timer indicating the remaining time for the voting campaign.
- **Casted Votes:** Display of the total number of votes casted by all users.
- **Candidates:** List of candidates running for the mayor position, including their names, ages, political parties, and photos.
- **Vote:** Users can vote for their preferred candidate through the DApp.
- **Winner Declaration:** Once the voting campaign ends, the contract owner can declare the winner, and the winner's name is displayed to all users.

# Smart Contract: Constructor & Main Functions

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.18;
3
4 contract Ballot {
5     struct Candidate {
6         uint256 id;
7         string firstName;
8         string lastName;
9         string party;
10        uint256 age;
11        string partyPhoto;
12        string candidatePhoto;
13        uint256 voteCount;
14    }
15
16    Candidate[] public candidates;
17    mapping(address => bool) public hasVoted;
18    address public contractOwner;
19    uint256 public campaignStartDate;
20    uint256 public campaignEndDate;
21
22    event VoteCast(address indexed voter);
23    event WinnerDeclared(uint256 indexed candidateId, uint256 voteCount);
24    event CandidateAdded(uint256 indexed candidateId, string firstName, string lastName);
25
26    modifier onlyOwner() {
27        require(msg.sender == contractOwner, "Only contract owner can perform this action");
28        _;
29    }
```

```
    constructor(Candidate[] memory _initialCandidates, uint256 _campaignEndDate) {
        contractOwner = msg.sender;
        campaignStartDate = block.timestamp;
        campaignEndDate = _campaignEndDate;

        for (uint256 i = 0; i < _initialCandidates.length; i++) {
            Candidate memory newCandidate = _initialCandidates[i];
            require(newCandidate.age >= 18, "Candidates must be at least 18 years old");
            newCandidate.id = i;
            candidates.push(newCandidate);
            emit CandidateAdded(newCandidate.id, newCandidate.firstName, newCandidate.lastName);
        }
    }

    function vote(uint256 _candidateId) public { ...
    }

    function declareWinnerId() public view onlyOwner returns (uint256) { ...
    }

    function announceWinner() public onlyOwner { ...
    }

    function getCandidatesCount() public view returns (uint256) { ...
    }


    function getVoteCount(uint256 _candidateId) public view returns (uint256) { ...
    }

    function getCandidateById(uint256 candidateId) public view returns (Candidate memory) { ...
    }
}
```



# Homepage

**VoteNow**

 [Connect Wallet](#)

89<sub>days</sub> 17<sub>hours</sub> 33<sub>min</sub> 50<sub>sec</sub>

Total Votes: 2


## Vote for the Mayor of Our City Make a Difference!

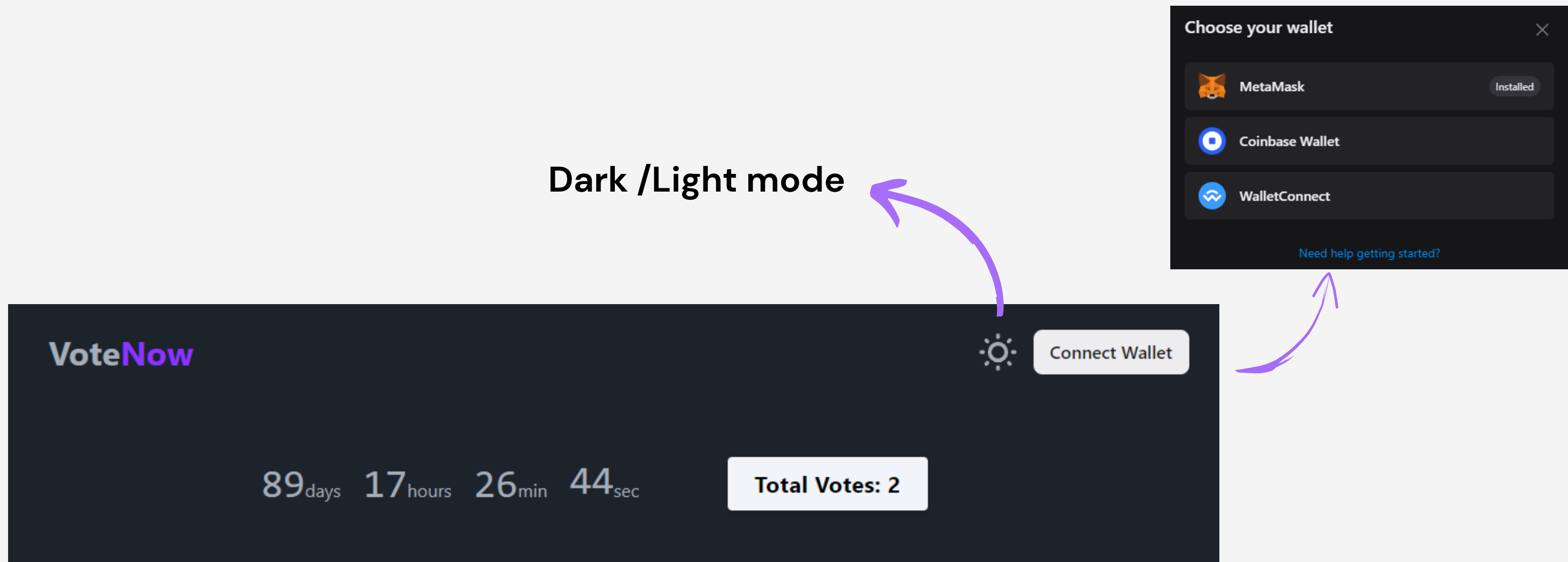
The crucial moment has arrived for our city! You have the opportunity to shape the future by voting for the next mayor. Connecting to Metamask is essential to participate in this important decision-making process.

The blockchain ensures the decentralization and transparency of the voting process, making it resistant to manipulation and fraud.

Join us and make a difference for our city!

See Candidates and cast your Vote!





Dark /Light mode

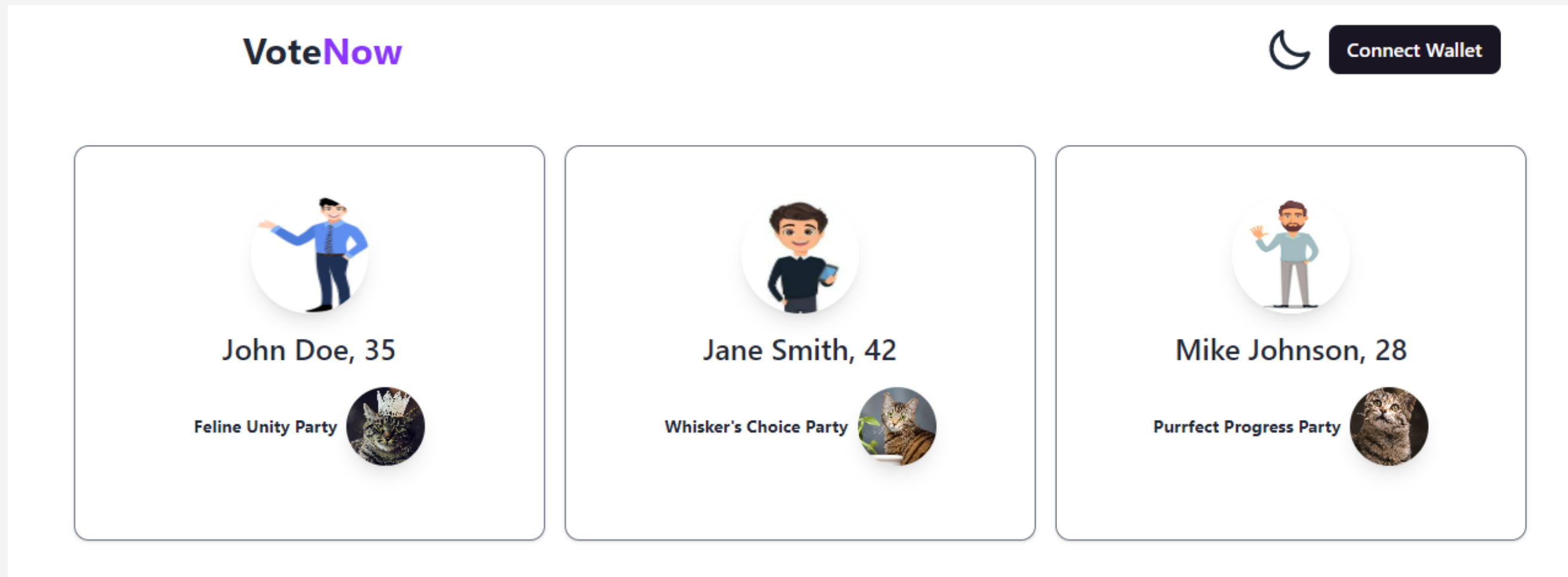
Duration of the voting campaign was set in the Smart Contract constructor

thanks to ether.js, the frontend listens to the "Vote Cast" event emitted by the Smart Contract.

The VoteCast event is triggered whenever a user successfully casts their vote for a candidate. With this feature, users can stay engaged and informed about the ongoing voting progress, creating a sense of transparency and trust in the voting system.

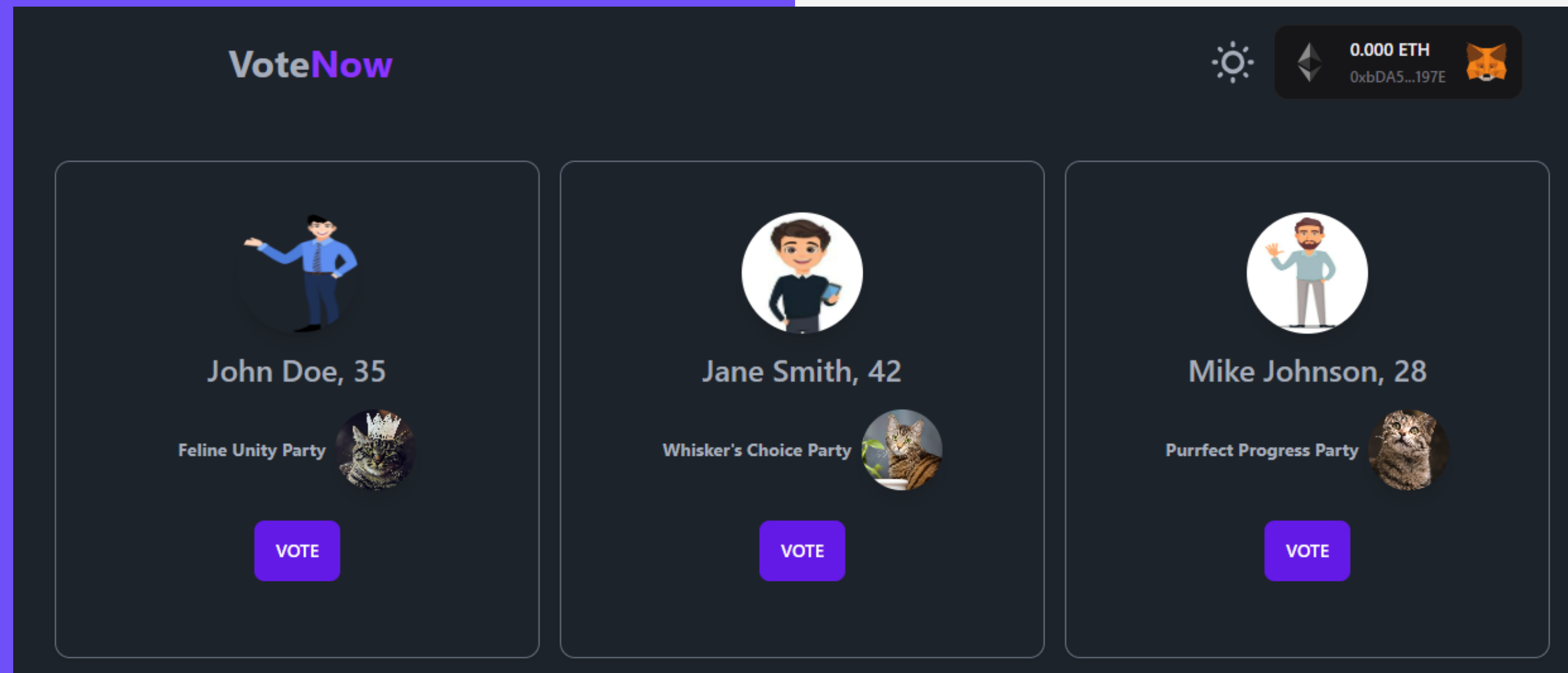
See Candidates and cast your Vote!

If you don't connect your wallet, you won't be able to see the vote button! The candidate data, like the campaign end date, were passed to the constructor during the creation of the SC.



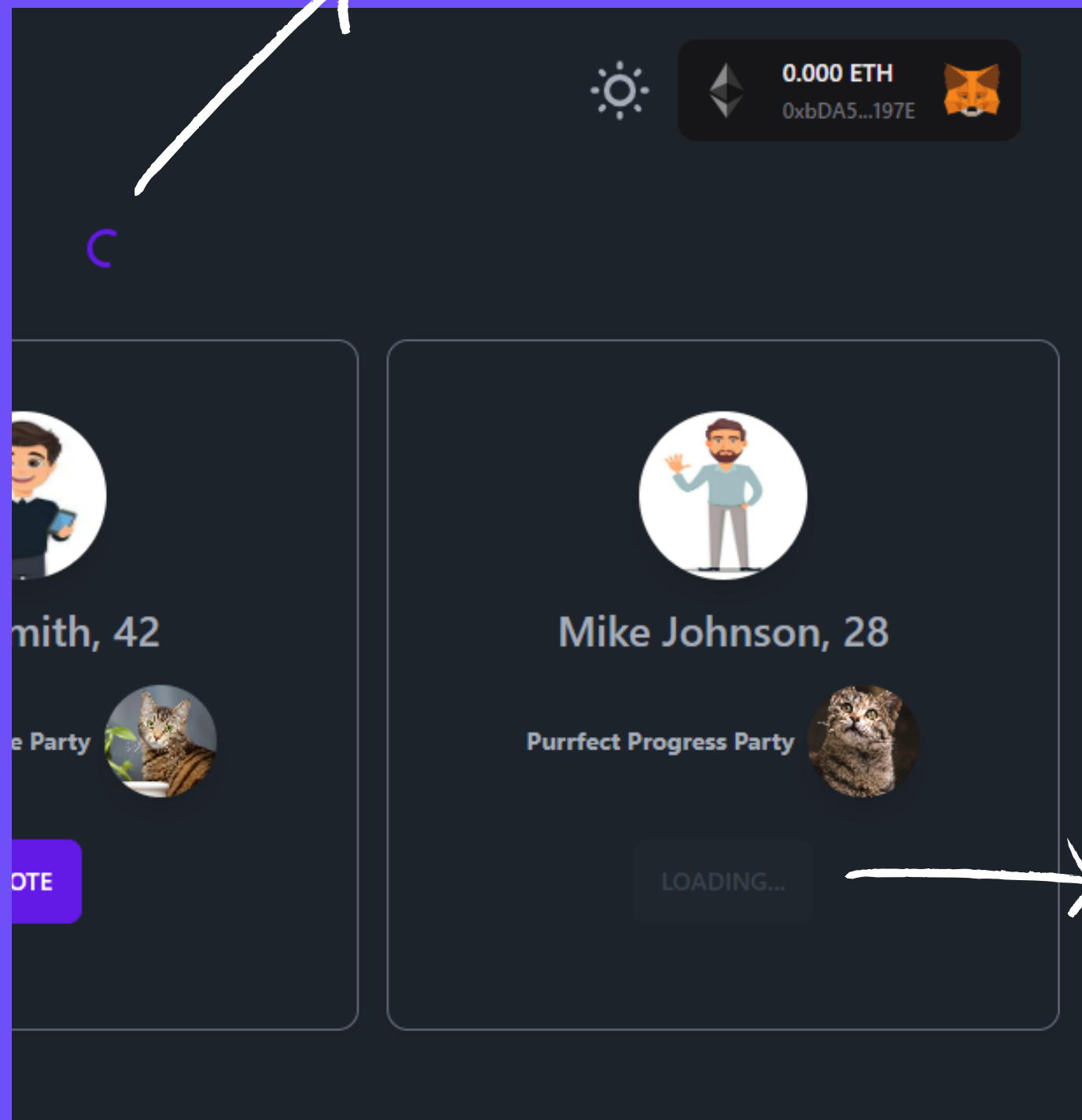
And as you can see, in this strange city political parties are based on... cats! 🐱



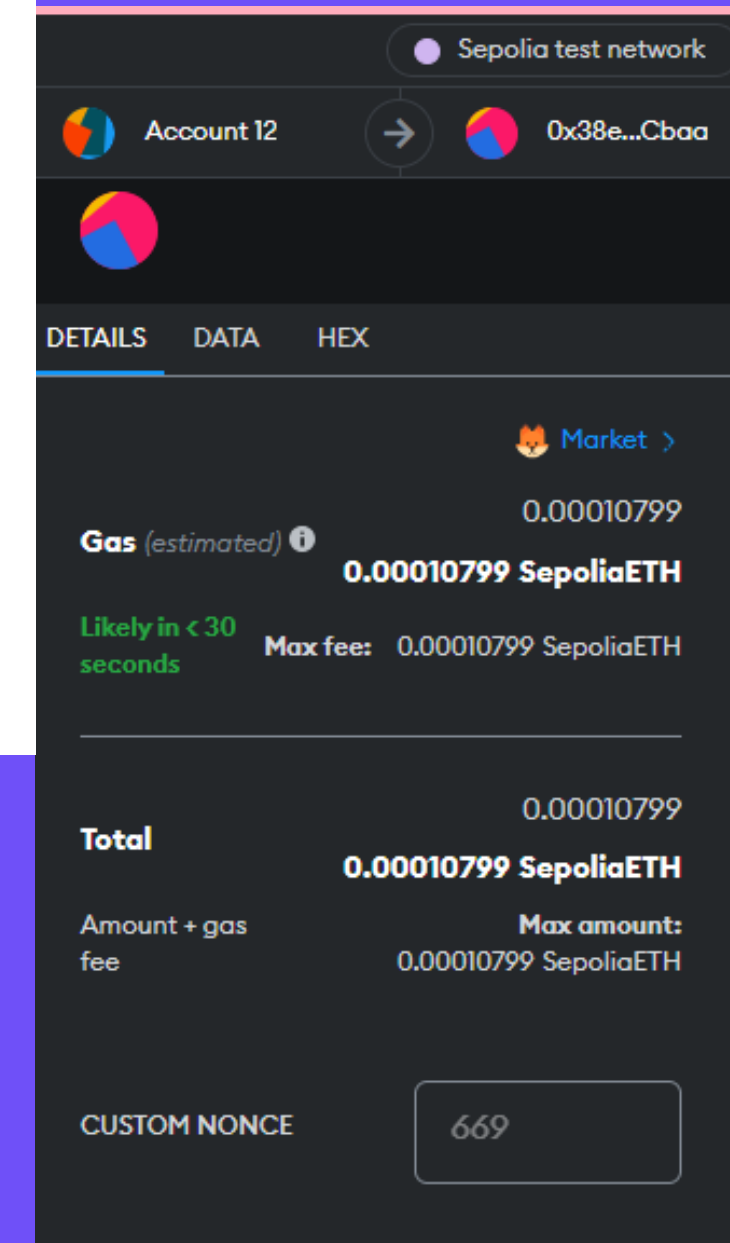
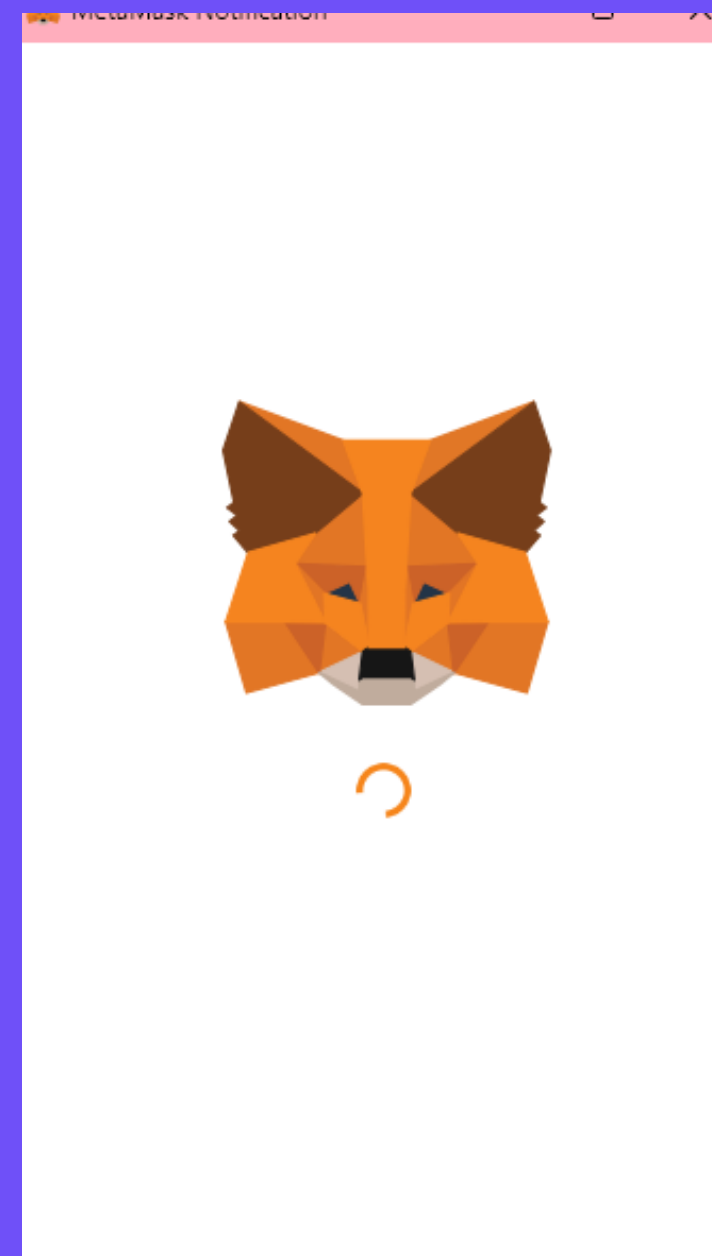


Now that you are connected to a wallet, you can proceed with the vote. You can only vote once, after which you will no longer be allowed to do so.

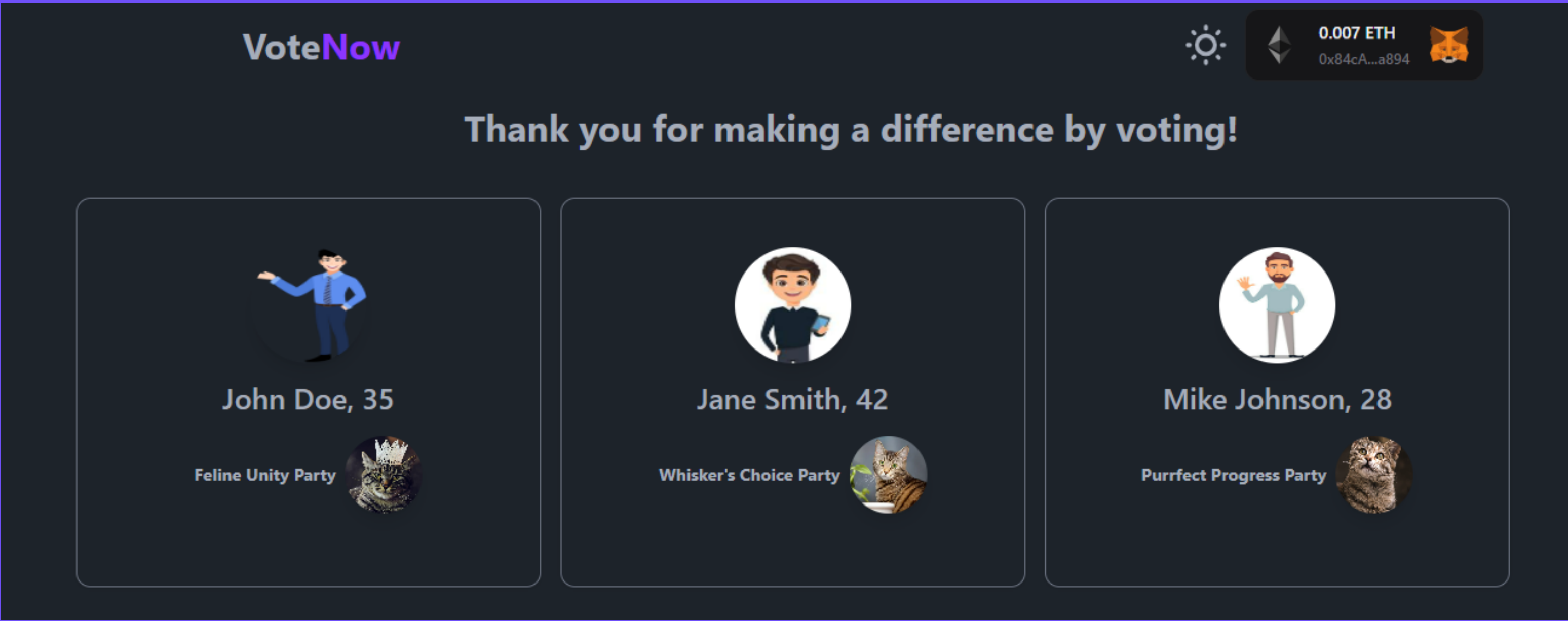
Loader during  
transaction's execution



Loader during  
transaction's execution



Once you have voted, the vote button will be disabled and a thank-you message will appear



# Contract Owner – 1



The contract owner has additional functionalities compared to a regular address. Firstly, they can view the real-time progress of votes for each candidate.

# Contract Owner – 2

At the end of the campaign, the contract owner will be enabled with a WinnerButton that allows calling the announceWinner function from the smart contract and declare the new mayor of the city!

```
const { mutateAsync: declareWinner } = useContractWrite(contract, "announceWinner")
let winnerName = "";

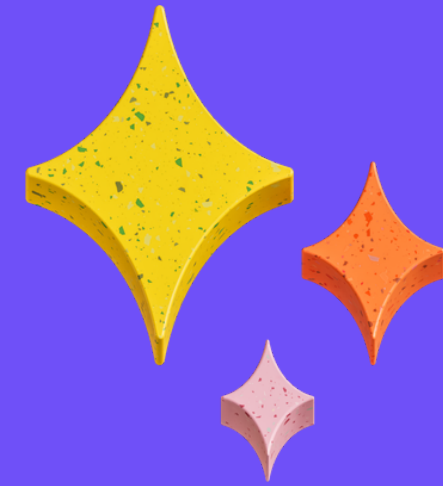
const handleCampaignEnd = () => {
  setIsCampaignEnded(true);
};

const handleDeclareWinner = async() => {
  try {
    const data = await declareWinner();
    console.info("contract call successs", data);
  } catch (error) {
    console.error("contract call failure", error);
  }
  setIsWinnerDeclared(true);
};

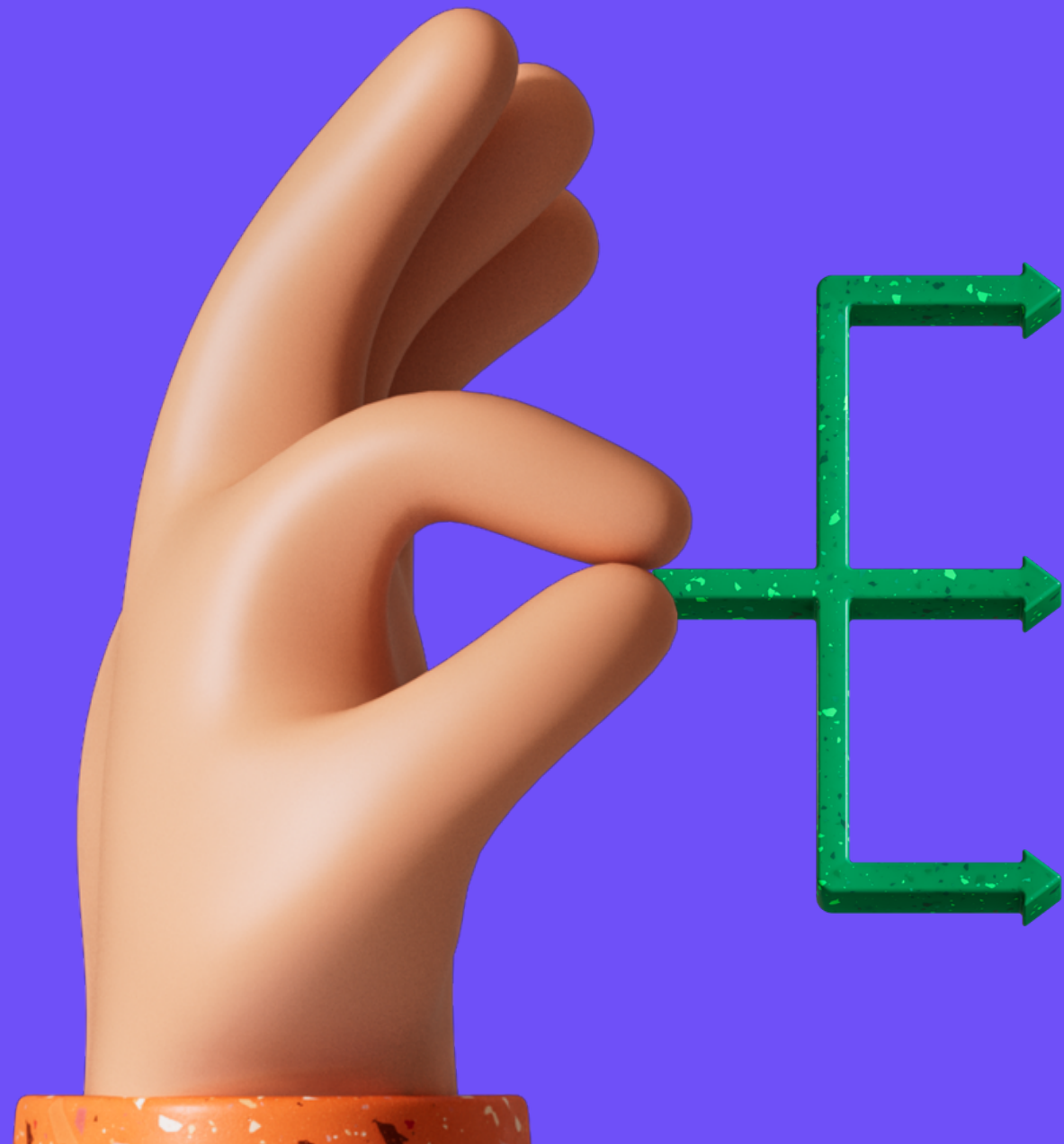
useEffect(() => {
  const fetchWinnerName = async () => {
    if (isCampaignEnded && isWinnerDeclared && contract) {
      const winnerId = await contract.call("declareWinnerId");
      const winner = await contract.call("getCandidateById", [winnerId]);
      winnerName = `${winner.firstName} ${winner.lastName}`;
      setIsWinnerDeclared(true);
    }
  };
  fetchWinnerName();
}, [isCampaignEnded, isWinnerDeclared, contract]);
```



# THANK YOU!



## Contacts



### Email

[diego.boost@gmail.com](mailto:diego.boost@gmail.com)

### GitHub

<https://github.com/diegoddie>

### Personal website

Coming soon🕒

**Diego Lauricella**