

Mini Project

AIM : Case study OR Mini Project on blockchain technology.

Project : Blockchain-Based E-Voting System

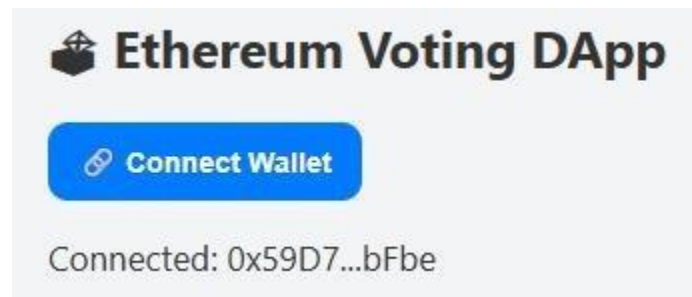
Introduction :

Electronic voting, or e-voting, is an emerging technology that digitizes the traditional voting process using secure, electronic systems. Traditional voting methods often face challenges like fraud, lack of transparency, and centralized control. Blockchain technology addresses these issues by providing decentralization, immutability, and transparency, making it an ideal candidate for e-voting applications.

This project implements an e-voting system on the Ethereum blockchain, allowing eligible voters to cast votes securely using smart contracts. Each voter can commit their vote and reveal it later, ensuring vote privacy and preventing double voting. The system leverages a commit- reveal scheme, a common cryptographic technique in blockchain-based voting, to ensure fairness and confidentiality.

Key features include:

- Only eligible voters can participate
- Votes are immutable once committed
- Transparent results verified on-chain
- Phased election process: Registration → Commit → Reveal → Ended



Case Studies :

Sr No	Title	Overview	Link
1	FollowMyVote	A U.S.-based blockchain voting platform enabling secure online elections with verifiable results	https://followmyvote.com
2	Horizon State	A blockchain voting platform that provides tamper-proof elections and citizen engagement	https://horizonstate.com
3	Voatz	Mobile voting application for secure voting in U.S. elections, leveraging blockchain and biometrics	https://voatz.com
4	West Virginia Blockchain Voting Pilot	Pilot program using blockchain to enable secure absentee voting for overseas military personnel	https://sos.wv.gov

Motivation :

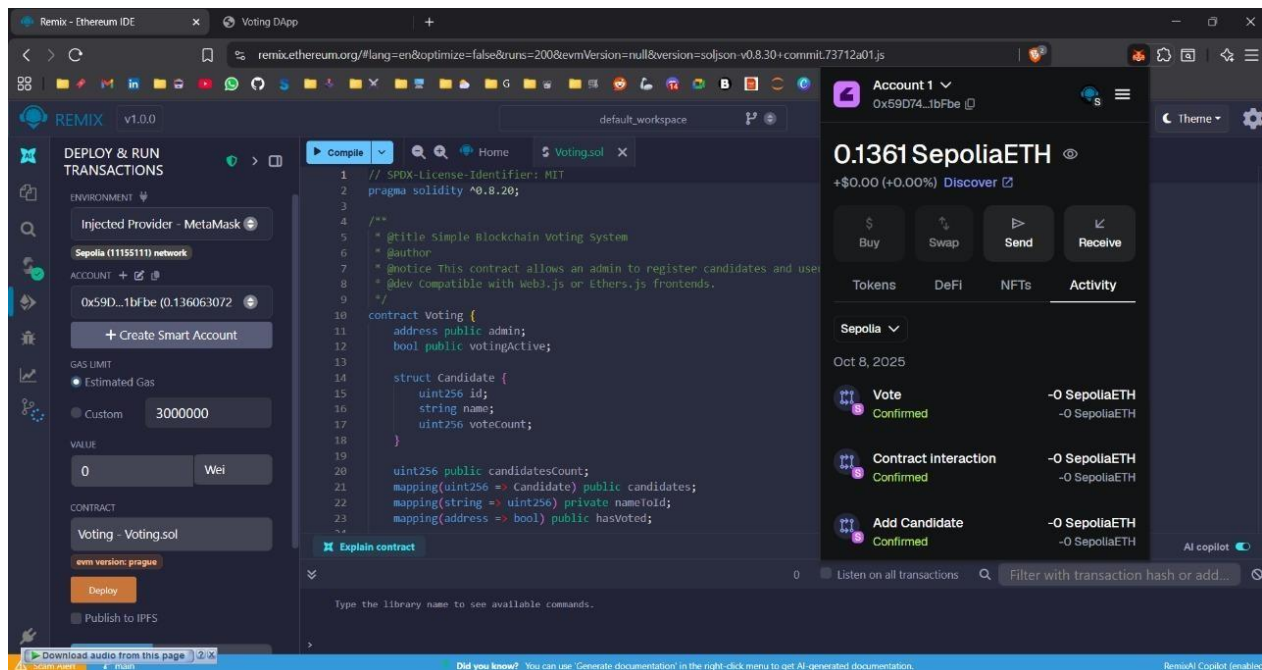
Traditional voting systems face multiple challenges:

- **Centralized control:** Results can be manipulated by administrators
- **Fraud risk:** Paper ballots are prone to tampering
- **Low accessibility:** Remote voters often struggle to participate

Blockchain-based e-voting addresses these issues by:

- Decentralizing vote storage
- Enforcing transparency via public ledger
- Using cryptographic techniques to secure voter privacy
- Automating processes through smart contracts

This project aims to **provide a secure, transparent, and fair voting system** using Ethereum smart contracts and Metamask integration, even without real ETH transactions.



Architectural Flow Diagram :

The e-voting system operates in **four phases**:

1. Registration Phase:

- Owner adds eligible voters to the system.

2. Commit Phase:

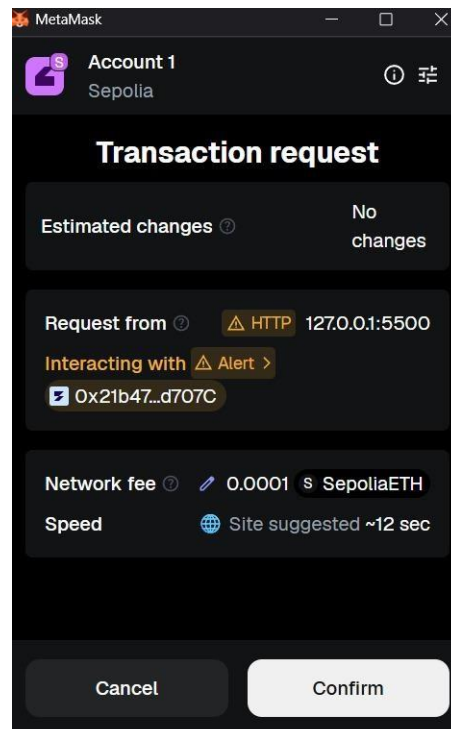
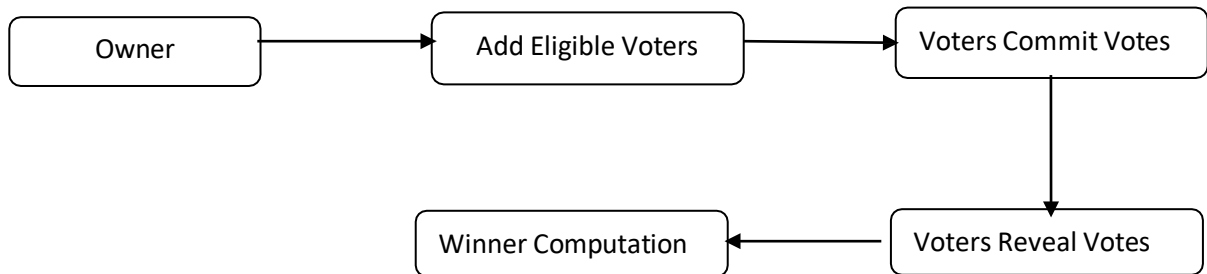
- Voters submit hashed votes (commitments) to maintain secrecy.

3. Reveal Phase:

- Voters reveal their votes and secrets, which are verified against the commitments.

4. Ended Phase:

- The contract computes and announces the winner based on revealed votes.



Implementation :

The smart contract EVoting.sol is written in Solidity for the Ethereum blockchain. It consists of:

Key Components:

- Ownable: Handles ownership and permissions
- Candidate struct: Stores candidate names and vote counts
- Phase enum: Tracks election phases
- eligible mapping: Tracks eligible voters
- commitments mapping: Stores vote hashes
- revealed mapping: Tracks which voters revealed votes

Key Functions:

1. Phase Control:

```
function startCommitPhase() external onlyOwner inPhase(Phase.Registration)
```

```
function startRevealPhase() external onlyOwner inPhase(Phase.Commit)
```

```
function endElection() external onlyOwner inPhase(Phase.Reveal)
```

2. Voter Registration:

```
function addVoter(address _voter) external onlyOwner
```

```
function addVoters(address[] calldata _voters) external onlyOwner
```

3. Voting (Commit & Reveal):

```
function commitVote(bytes32 _commitment) external
```

```
function revealVote(uint _candidateIndex, string calldata _secret) external
```

4. Winner Calculation:

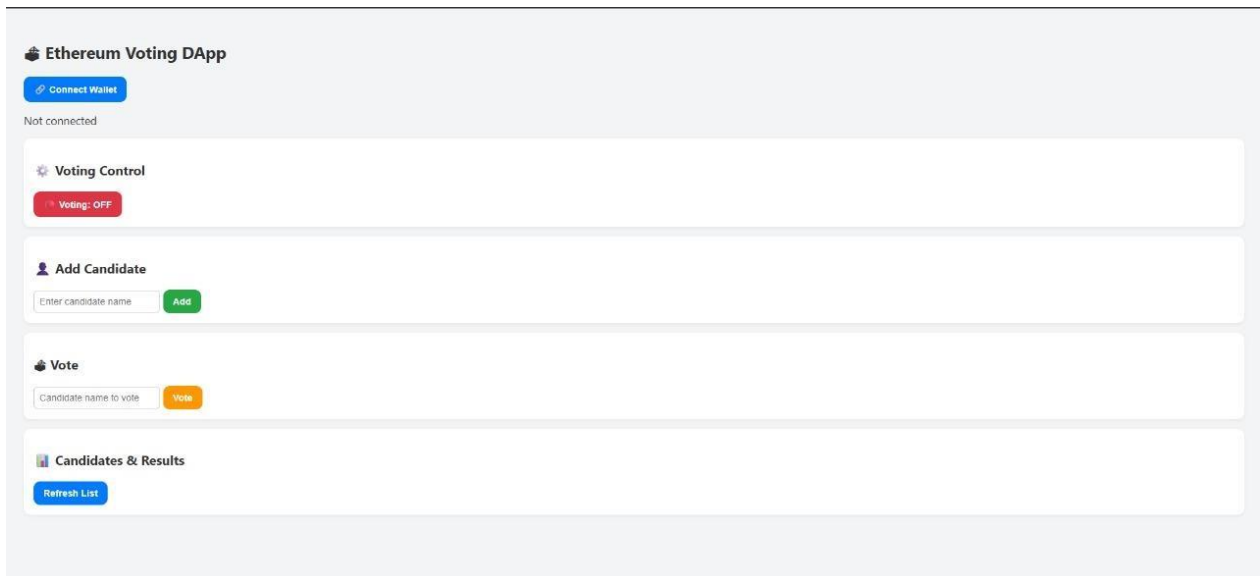
```
function winner() external view inPhase(Phase.Ended) returns (...)
```

Process:

- Voters compute a commitment using $\text{keccak256}(\text{candidateIndex} + \text{secret})$
- Commitments are submitted during the Commit phase
- Votes are revealed in the Reveal phase, and the contract ensures integrity
- The winner is determined automatically by counting votes

Integration with Metamask:

- Users connect their Ethereum wallets
- Transactions (commit/reveal) are signed and submitted to the blockchain.



The screenshot shows the 'Ethereum Voting DApp' interface. At the top, there is a 'Connect Wallet' button and a status 'Not connected'. Below this, there are four main sections: 1. 'Voting Control' with a 'Voting: OFF' indicator. 2. 'Add Candidate' with an input field 'Enter candidate name' and an 'Add' button. 3. 'Vote' with an input field 'Candidate name to vote' and a 'Vote' button. 4. 'Candidates & Results' with a 'Refresh List' button.

Conclusion :

The blockchain-based e-voting system demonstrates:

- Secure and private voting using smart contracts
- Transparent, immutable vote recording
- Reduced fraud and manipulation risks
- Practical deployment on Ethereum testnets

Even without real ETH, the contract and architecture prove that a **decentralized, secure e-voting system** is feasible. Future enhancements could include:

- Integration with a web frontend
- Support for large-scale elections
- Improved voter authentication (e.g., biometric verification)

References :

1. Ethereum Smart Contracts: <https://ethereum.org/en/developers/docs/smart-contracts/>
2. Solidity Documentation: <https://docs.soliditylang.org/>
3. FollowMyVote: <https://followmyvote.com>
4. Horizon State: <https://horizonstate.com>
5. Voatz: <https://voatz.com>
6. West Virginia Blockchain Voting Pilot: <https://sos.wv.gov>