Cover letter

Blockchain-based Electronic Voting Machine

Table of Contents

ABSTRACT:

The EVS (Electronic Voting System) is a method of conducting elections which uses blockchain technology and which is designed to be secure, transparent and immutable and therefore able to revolutionize the current electoral process. The EVS utilizes blockchain's decentralized and cryptographically secure functions to prevent votes being counted in error and makes it possible for all parties to check if votes were counted accurately at any time. In the EDS system, voters can also vote from home using their computer or mobile device and the technology is designed in a manner that guarantees safety and security still. The blockchain system ensures electoral process transparency and eliminates the risk of tampering or fraud by recording each vote as a transaction.

The growth of EDS has seen a step-change leading to the deeper integration of blockchain technology. Major parts of system architecture are being developed, and basic features like voter authentication processes have been implemented. Blockchain technology has already been employed in the tallying and the submission of the votes for example in the 2018 midterm elections by the California officials. Executing the evaluation process paying great attention to the system security, usability, and scalability, the tests and validation are being made. In the near future our team will be working to strengthen the security system, improve its user-interface, and run pilot-testing to check if it is possible that the system can meet our expectations in its efficient functioning. After EDS get implemented on blockchain technology organization has gone through some crucial modifications in order to enhance the whole system and security capabilities. The use of the latest encryption technologies enabling the protection of vote transmission over the internet and preventing from vote breaching or interception can be considered as a significant achievement. What is more, the current voting system is a decentralized one and all transactions are recorded leaving a trail on the blockchain while making all elections transparent and verifiable. Moreover, EDS is decisively scalable and equally capable of dealing with almost all elections that include millions of votes. In the light of all the arguments, we can state that the mentioned improvements have made EDS based on the blockchain a more reliable, transparent, and effective but the democratic process safe platform.

INTRODUCTION:

In the modern world, data is regarded as the most asset a person or organization can have. Different organizations utilize various types of data to solve challenges. The data they possess is their greatest asset. Organizations across the world are doing everything they can to protect their data because of this. Since 2009, the use of blockchain technology to secure data has become essential. It guarantees the integrity, validity, and anonymity of data. A blockchain's data can't currently be altered, erased, corrupted, etc. it is a cutting-edge platform that offers safe, transparent, and uncorruptible system. Using a decentralized and distributed ledger, this system provides an immutable record of each vote and shields it against manipulation or unauthorized access. Cryptographic techniques can be used to protect voter identities while preserving the transparency and accountability of the voting process. By utilizing blockchain technology, voting becomes more accessible, safe, and efficient, allowing for distant participation and

quicker, auditable outcomes. Trust, democratic values, and an inclusive election process are the ultimate goals of a blockchain-based electronic voting system.
Elections are conducted online. From a technical standpoint, it concentrates on:

- Election calling
- Candidate registration
- Polling place preparation
- Correct voting
- Vote counting
- Auditing, reviewing
- follow-up.

The scope of electronic voting can also include auditing, evaluating, and follow-up. Additionally, it is not the duty of electronic voting to organize campaigns or persuade voters in order to advance the interests of other individuals. Voting Setup, or the components that make up the e-voting system's input.
The output is made up of:

- Summarized voting results from e-voters, which include the voter's allocated polling division and constituency
- Candidate list
- The expressed will of the voters.
- An electronic voting user list. There are all the necessary components and participants in place to administer the system.

LITERATURE REVIEW:

OVERVIEW :

Those elections having automated voting machines run smoothly and can be carried out more effectively, but it is essential for those machines to be secure ,free and fair. Being immutable and decentralized, blockchain technology has demonstrated the story which can be a radical resurgence to these issues. Electronic Voting Machines can assist in a secure process through use of blockchain technology that provides transparency and a guarantee of confidence.

BACKGROUND:

In modern day world of state of art elections with the use of modern automated voting machine, the system has gained more enhancement and efficiency, consequently, reliability and effectiveness. Although the main factor is still the security and integrity of the systems, this elections should be conducted in an open and orderly manner that would not attract doubt of unethical practices. Now the blockchain technology comes with a great innovation which is a solution to the problems we face. With its complete non-alterable and uncentralized nature, as well as good data quality preservation features, blockchain is a great book of proof how it can be the deciding factor in the future of electoral world. Utilize of the blockchain and the employment for the electronic voting system not just the resources of security and transparency can be provided as well as the trust on the confidentiality of the electoral process can be reached.

The use of blockchain technology in electronic voting systems is a real breakthrough that will yield a number of benefits such as improved security, transparency, and integrity. With the decentralization process, the system has a higher resistance to possible attacks, and it helps to keep the integrity of the entire electoral process. Blockchain's innate strengths provide the capability of keeping records that cannot be tampered with or unlawfully modified, which increases transparency, safety, and decreases the chances of fraud. Smart contracts likewise improve efficiency by automating the enforcement of rules and the voting procedures, which makes the electoral process easier and without compromising the integrity and accountability.

Although electronic voting systems using blockchain technology are highly probable, they also left with some difficulties. Scaleability is the most concerning effects seeing that ample number of votes need level 2 scaling plugins on the whole the chain. In addition to this, we also have to face the problem of user adoption and key management, which are the practical challenges that must be solved to make sure that the usage and usability of the system is widespread. The implementation of blockchain in electronic voting also ascertains that there are some complications, but at the same time, this confirms that it is not almost the best available of its kind and we should do more research and development to unmistakably get to the full potential of blockchain technology when it comes to the transformation of electoral mechanisms.

USING BLOCKCHAIN TECHNOLOGY TO ELECTRONIC VOTING:

In addition to electronic voting systems, blockchain technology avails security, transparency, and integrity. Decentralization of the system results in enhanced network safety. Therefore, blockchain technology is a great tool for electronic voting systems. It ensures the election' s honorable nature and a failure-proof records keeping and voting method. By intelligently deploying smart contracts to automate rule enforcement and voting, the proceeding steps end up becoming efficient.

INTEGRITY AND SECURITY IN ELECTRONIC VOTING:

Blockchain records votes in an unchangeable and open way, while keeping votes and transactions encrypted to provide security and keep the electronic voting systems uncorrupted. This also eliminates the possibility for one' s vote to be falsely tampered with or omitted in the counting process since it was clearly cast. Secondly, it disapproves of deceit, foul play, and unlawful entry.

TRANSPARENCY AND DECENTRALIZATION:

The feature of the blockchain movement which is " decentralized" , thus, makes the election process less prone to manipulation because no one party can control the voting mechanism electronically. Being open about the paper trail of the voting process, everyone will be able to know whether it is not bias or favoritism, encouraging all system to have confidence about election result.

OBSTACLES AND RESTRICTIONS:

The benefits of the electronic voting systems based on the Blockchain include scalability. However, they also suffer from shortcomings, namely, the key management, and the user uptake.
A great number of votes needs scalability solutions, and the system might become too intricate for everyone to adopt the use of blockchain technique. To prevent unwanted beneficiaries from accessing voting accounts, keys storage should be made paramount.

CASE STUDIES AND THEIR APPLICATION:

The blockchain has been deployed in electronic voting systems in the real world on just a few occasions, and which cases were successful, and which not is difficult to tell. Within these applications they learned that this area need to be studied more and that blockchain comes with its own advantages and disadvantages and this opens up a manual of using this technology for e-voting purposes.

LEGAL AND REGULATORY CONSIDERATIONS:

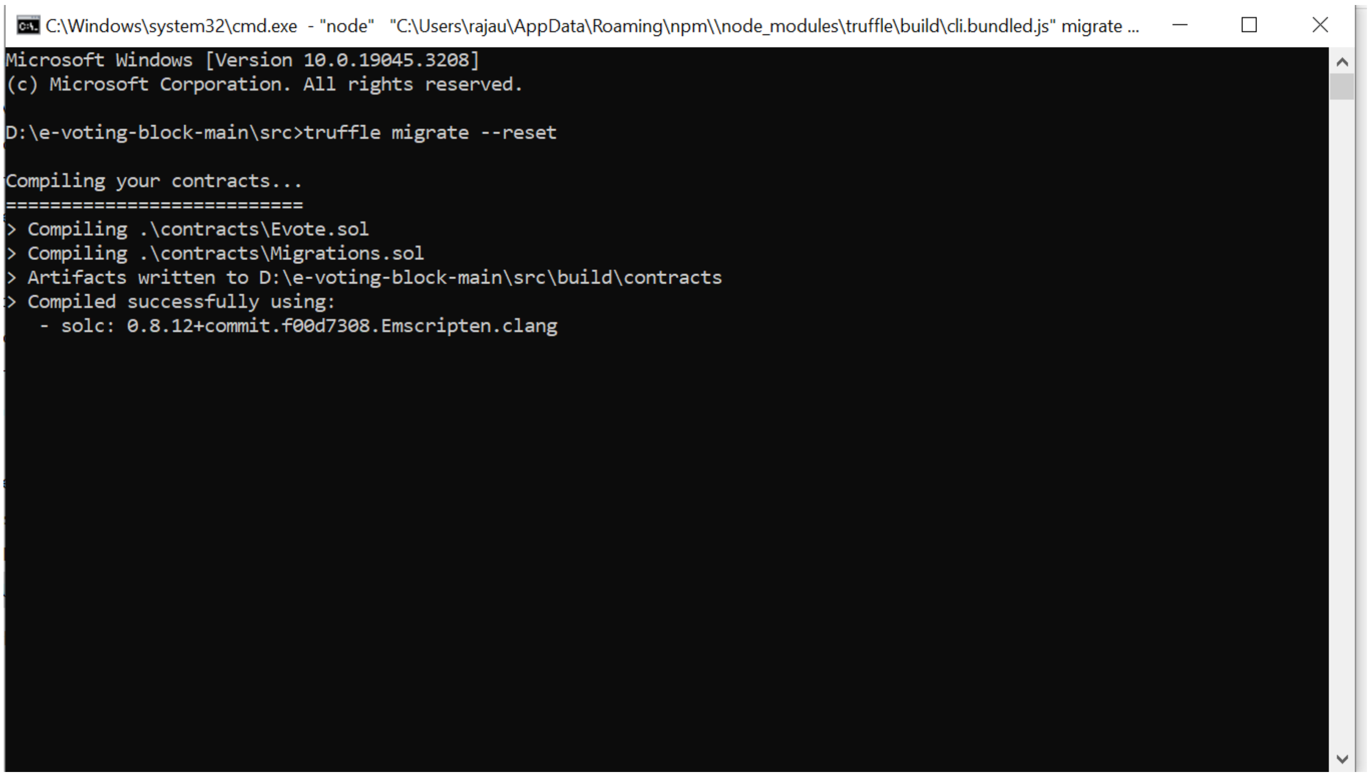
Blockchain technology can be applied to the voting systems of e-elections to go with the law on elections, privacy, and data protection. While the blockchain-based electronic voting systems are working to be up to the regulations, the main focus is to guard voter’ s privacy and to solve this.

Future guidelines: It is, and would be insightful to carry out research focusing on the elements like usability, security and scalability which will be blockchain-based electronic voting systems. In order to completely restructure elections and start operating them using blockchain technology, breakthroughs would be needed in some of the domains specified below.

IMPLEMENTATION:

1. SCREENSHOT OF CONFIGURATION:

MIGRATION EXECUTION



C:\Windows\system32\cmd.exe - "node" "C:\Users\rajau\AppData\Roaming\npm\\node_modules\truffle\build\cli.bundled.js" migrate ...

```
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.

D:\e-voting-block-main\src>truffle migrate --reset

Compiling your contracts...
===========================
> Compiling .\contracts\Evote.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to D:\e-voting-block-main\src\build\contracts
> Compiled successfully using:
   - solc: 0.8.12+commit.f00d7308.Emscripten.clang
```

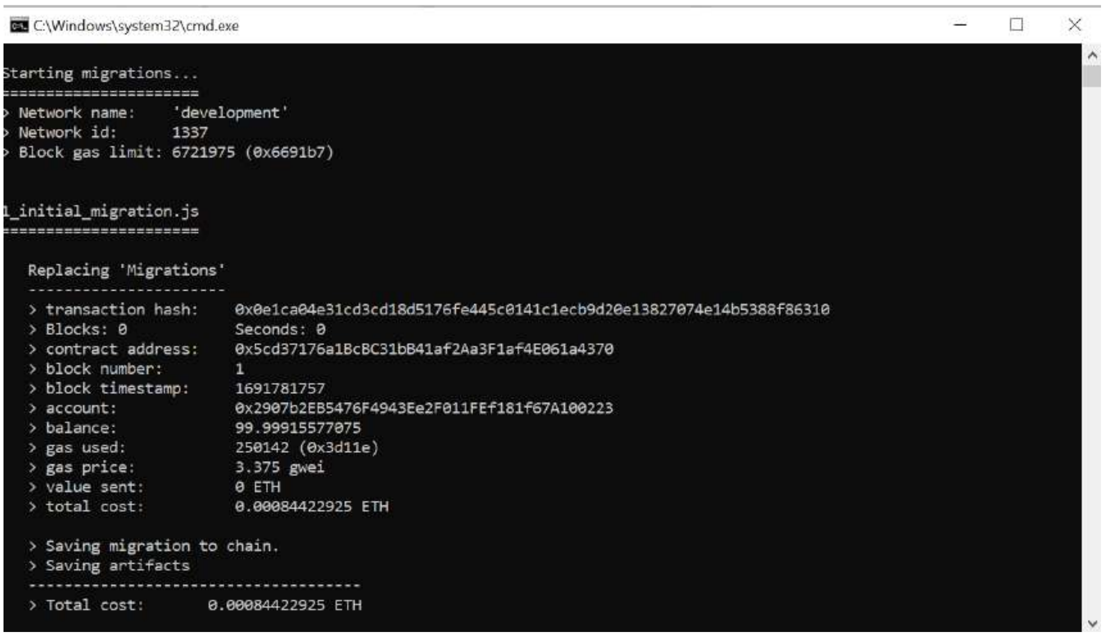
Figure 1 Migration

Here is the compilation of two smart contracts.

1. Contract Deployment:

1.1 Development of migration Smart Contract

Here is the migrations screenshot of Migrations.

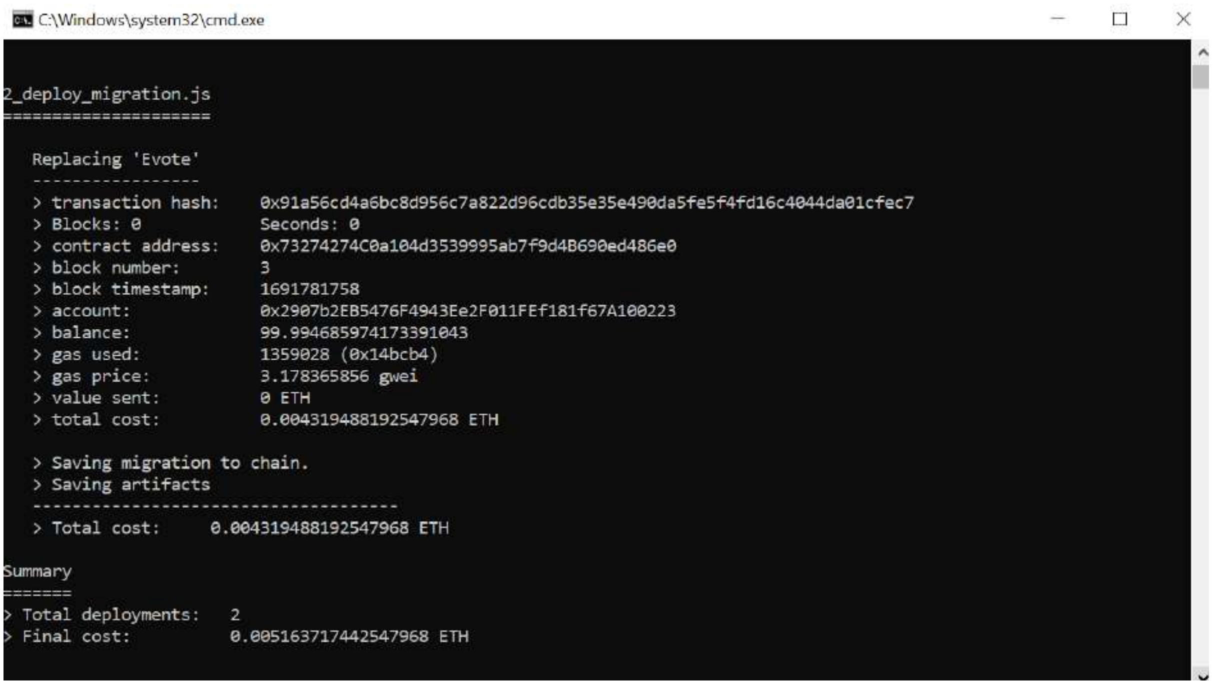


```
C:\Windows\system32\cmd.exe

Starting migrations...
======================
> Network name:    'development'
> Network id:      1337
> Block gas limit: 6721975 (0x6691b7)


1_initial_migration.js
======================

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0x0e1ca04e31cd3cd18d5176fe445c0141c1ecb9d20e13827074e14b5388f86310
   > Blocks: 0            Seconds: 0
   > contract address:    0x5cd37176a1BcBC31bB41af2Aa3F1af4E061a4370
   > block number:        1
   > block timestamp:     1691781757
   > account:             0x2907b2EB5476F4943Ee2F011FEf181f67A100223
   > balance:             99.99915577075
   > gas used:            250142 (0x3d11e)
   > gas price:           3.375 gwei
   > value sent:          0 ETH
   > total cost:          0.00084422925 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:       0.00084422925 ETH
```

Figure 2 Migration of Smart Contract

1.2 Deployment of E-Vote Smart Contract

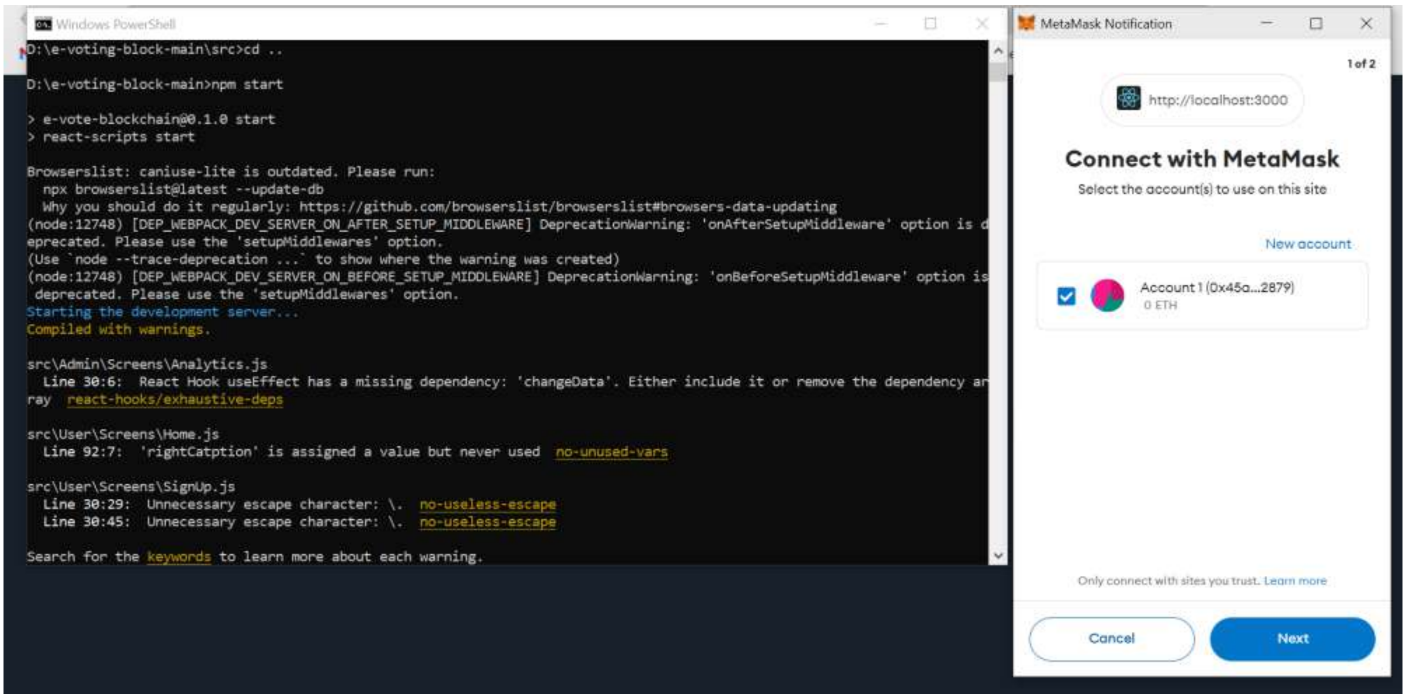
Here is the deployment of E-vote smart Contract.



```
C:\Windows\system32\cmd.exe


2_deploy_migration.js
=====================

   Replacing 'Evote'
   -----------------
   > transaction hash:    0x91a56cd4a6bc8d956c7a822d96cdb35e35e490da5fe5f4fd16c4044da01cfec7
   > Blocks: 0            Seconds: 0
   > contract address:    0x73274274C0a104d3539995ab7f9d4B690ed486e0
   > block number:        3
   > block timestamp:     1691781758
   > account:             0x2907b2EB5476F4943Ee2F011FEf181f67A100223
   > balance:             99.994685974173391043
   > gas used:            1359028 (0x14bcb4)
   > gas price:           3.178365856 gwei
   > value sent:          0 ETH
   > total cost:          0.004319488192547968 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:     0.004319488192547968 ETH

Summary
=======
> Total deployments:   2
> Final cost:          0.005163717442547968 ETH
```

Figure 3 Contract Deployment

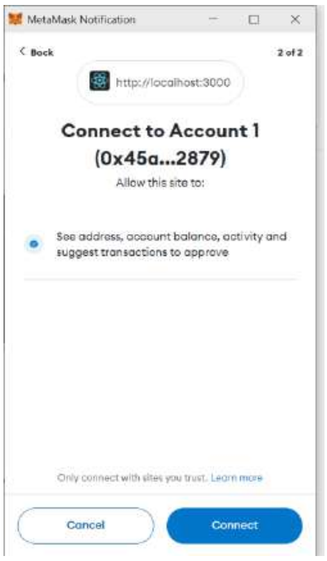
2. METAMASK CONNECTIONS



```
D:\e-voting-block-main\src>cd ..

D:\e-voting-block-main>npm start

> e-vote-blockchain@0.1.0 start
> react-scripts start

Browserslist: caniuse-lite is outdated. Please run:
  npx browserslist@latest --update-db
  Why you should do it regularly: https://github.com/browserslist/browserslist#browsers-data-updating
(node:12748) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is d
eprecated. Please use the 'setupMiddlewares' option.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:12748) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is
 deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled with warnings.

src\Admin\Screens\Analytics.js
  Line 30:6:  React Hook useEffect has a missing dependency: 'changeData'. Either include it or remove the dependency ar
ray  react-hooks/exhaustive-deps

src\User\Screens\Home.js
  Line 92:7:  'rightCatption' is assigned a value but never used  no-unused-vars

src\User\Screens\SignUp.js
  Line 30:29:  Unnecessary escape character: \.  no-useless-escape
  Line 30:45:  Unnecessary escape character: \.  no-useless-escape

Search for the keywords to learn more about each warning.
```

Figure 4 MetaMask Connections

Here is connection of ganache network on MetaMask. Ensure that Ganache is operational and set up to utilize the same network before connecting the Ganache network to MetaMask. Choose "Custom RPC" from the network drop-down menu at the top of MetaMask. Enter the Ganache network information, including the string ID (preferably 1337 for Ganache if it is possible) and the RPC server URL (often http://localhost:7545). To incorporate the Ganache network into MetaMask, click "Save". Once installed, you should be able to access and interact with the locally running Ganache blockchain using your Ganache accounts in MetaMask


MetaMask Notification
< Back
2 of 2
http://localhost:3000
Connect to Account 1 (0x45a...2879)
Allow this site to:
See address, account balance, activity and suggest transactions to approve
Only connect with sites you trust. Learn more
Cancel
Connect

Here is confirmation message that your ganache network account is successfully connected.

NEW WORKSPACE:

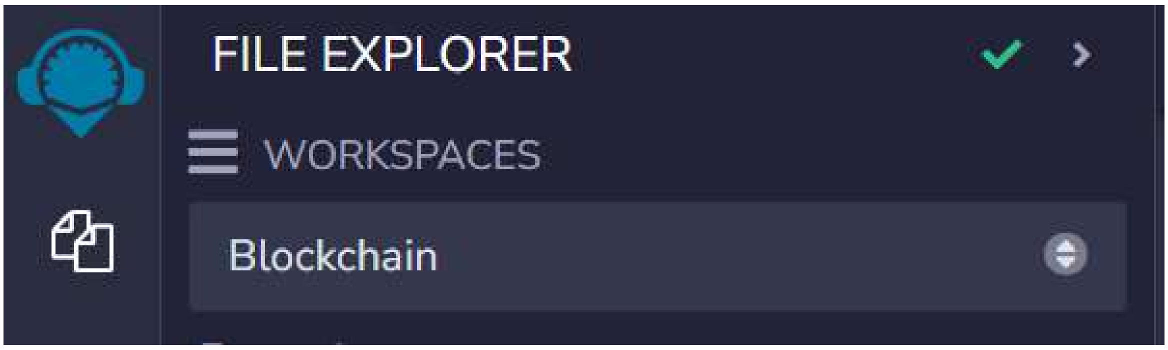
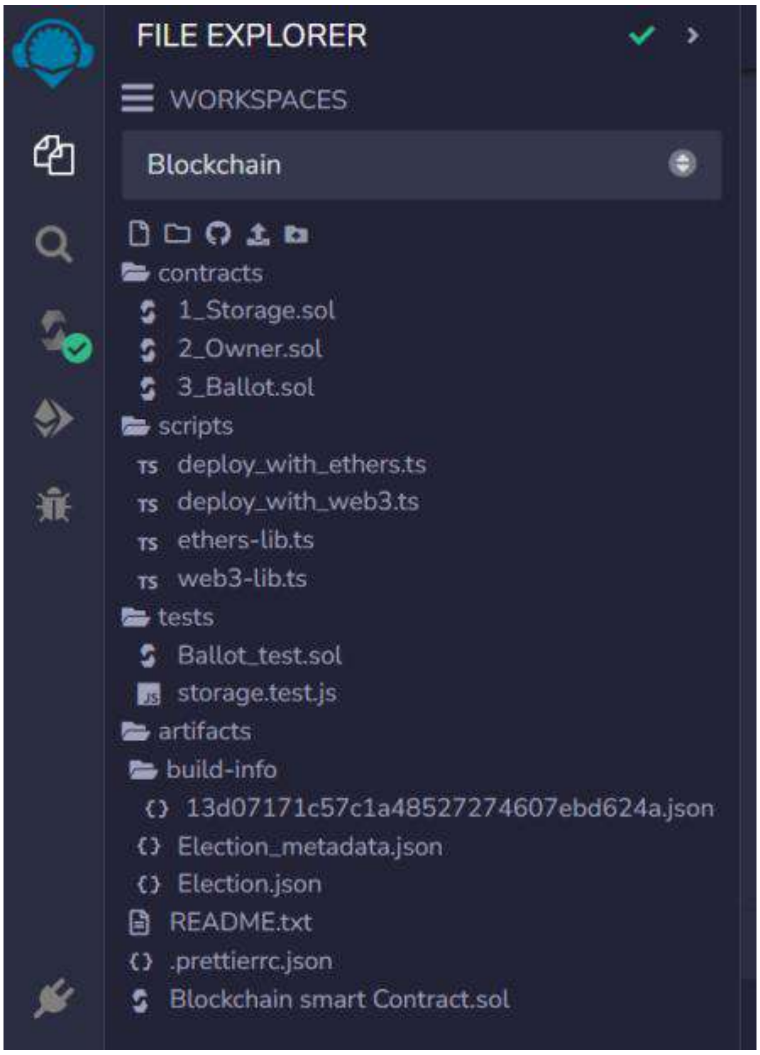


Figure 5 Remix Workspace

CREATION OF NEW FILE:

Creating a new file:

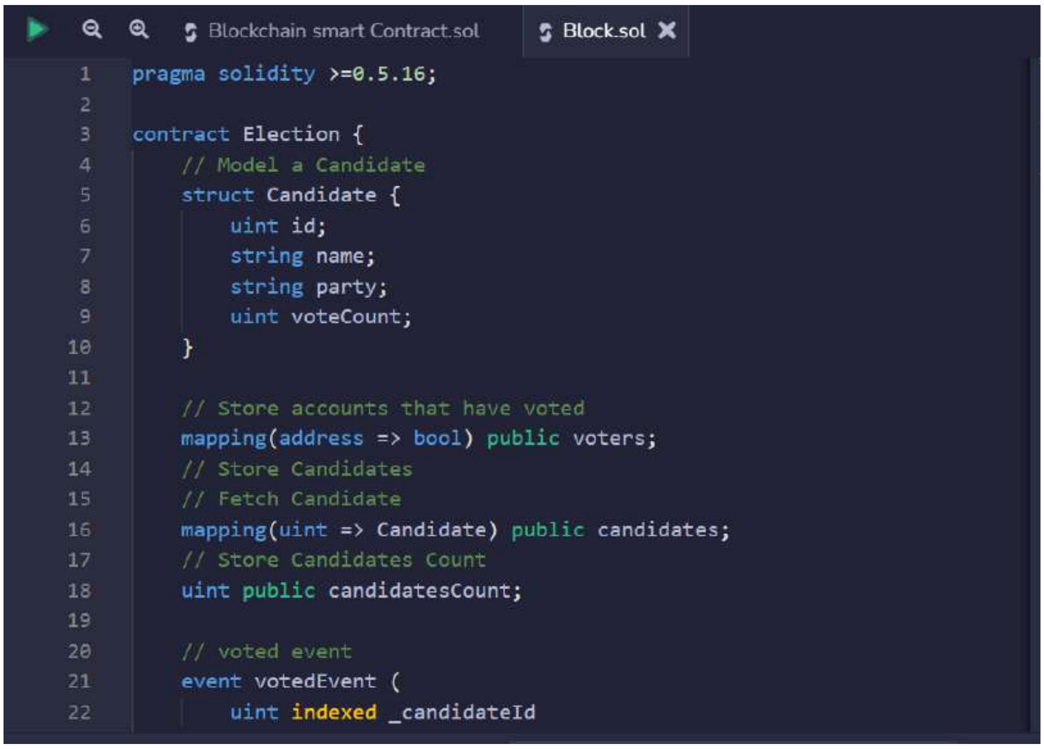

FILE EXPLORER
WORKSPACES
Blockchain

```
contracts
  1_Storage.sol
  2_Owner.sol
  3_Ballot.sol
scripts
  deploy_with_ethers.ts
  deploy_with_web3.ts
  ethers-lib.ts
  web3-lib.ts
tests
  Ballot_test.sol
  storage.test.js
artifacts
  build-info
    13d07171c57c1a48527274607ebd624a.json
  Election_metadata.json
  Election.json
README.txt
.prettierrc.json
Blockchain smart Contract.sol
```

Figure 6 New File of Solidity Smart Contract

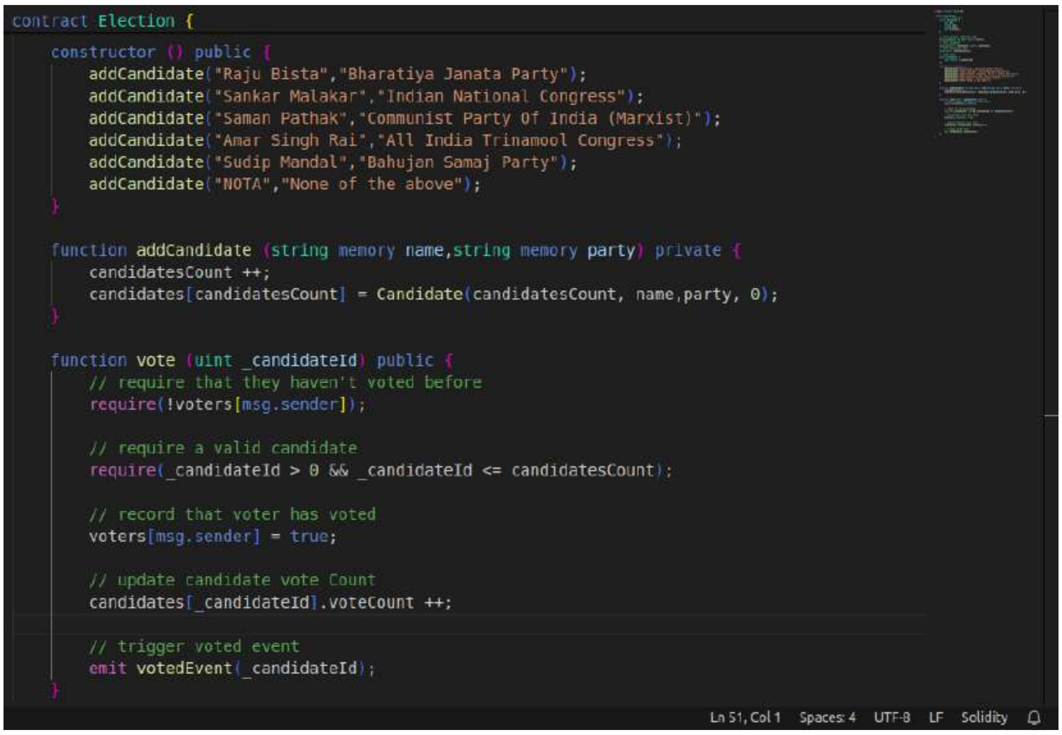
CREATION OF SMART CONTRACT:

Electronic voting on the blockchain is made possible by this Solidity smart contract, called the "E-Voting Machine". Voter and candidate registration is made possible by it. Voters can register their addresses and add candidates. Voters are able to select candidates after registering. The agreement includes a mechanism to end the voting process and prohibits duplicate voting. Lastly, by identifying the candidate who received the most votes, the contract may ascertain the outcome of the election.



```
pragma solidity >=0.5.16;

contract Election {
    // Model a Candidate
    struct Candidate {
        uint id;
        string name;
        string party;
        uint voteCount;
    }

    // Store accounts that have voted
    mapping(address => bool) public voters;
    // Store Candidates
    // Fetch Candidate
    mapping(uint => Candidate) public candidates;
    // Store Candidates Count
    uint public candidatesCount;

    // voted event
    event votedEvent (
        uint indexed _candidateId
```

Figure 7 Block Contract of Solidity



```
contract Election {
    constructor () public {
        addCandidate("Raju Bista","Bharatiya Janata Party");
        addCandidate("Sankar Malakar","Indian National Congress");
        addCandidate("Saman Pathak","Communist Party Of India (Marxist)");
        addCandidate("Amar Singh Rai","All India Trinamool Congress");
        addCandidate("Sudip Mondal","Bahujan Samaj Party");
        addCandidate("NOTA","None of the above");
    }

    function addCandidate (string memory name,string memory party) private {
        candidatesCount ++;
        candidates[candidatesCount] = Candidate(candidatesCount, name,party, 0);
    }

    function vote (uint _candidateId) public {
        // require that they haven't voted before
        require(!voters[msg.sender]);

        // require a valid candidate
        require(_candidateId > 0 && _candidateId <= candidatesCount);

        // record that voter has voted
        voters[msg.sender] = true;

        // update candidate vote Count
        candidates[_candidateId].voteCount ++;

        // trigger voted event
        emit votedEvent(_candidateId);
    }
```

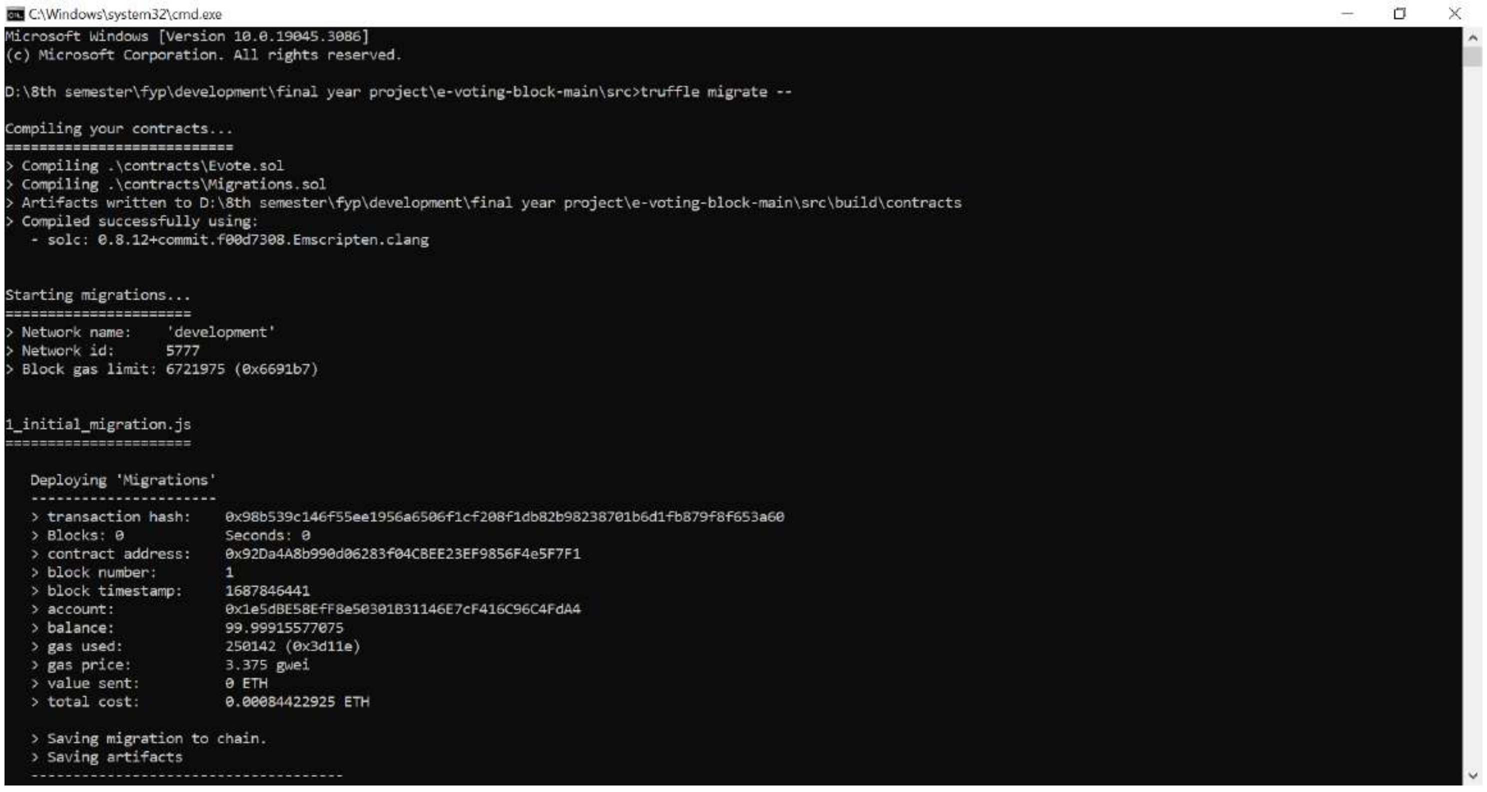
The voting use-case is inserted into the Solidity smart contract through specific implementation stages, where voters can choose candidates. creates an "Cand" form to keep track of every candidate associated with information including their name, candidate, and total number of votes, respectively. Voters agreeing to commitments assure the integrity of their contributing vote and also of the candidate they support additionally they are voting for. First, the contract checks the eager voters wish to participate using the voter ID, and then it follows up with the advanced poll voters, whose addresses have been tracked using mapping. The contract entails the recording of the ballot, a rise of the candidate's vote count, and emits an output to alert the new status if the conditions have been met. In brief, this contract provides incompetent procedure for running elections on the Ethereum platform being straightforward but useful at the same time.

1. TESTING:

We need to open ganache application for migration truffle. Then we need to open its src folder in cmd and write truffle migrate command.

Screen shot for truffle configuration and migration:

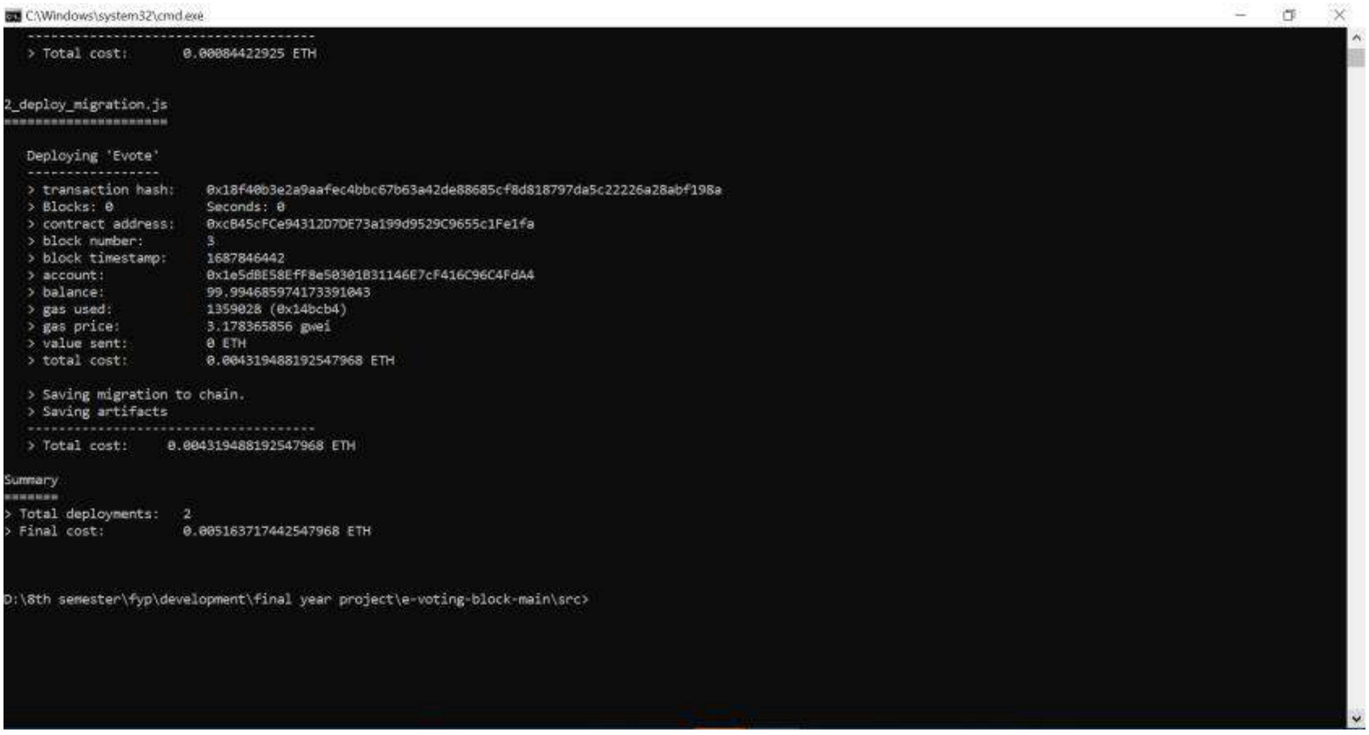
1.1 TRUFFLE MIGRATE SCREENSHOT



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

D:\8th semester\fyp\development\final year project\e-voting-block-main\src>truffle migrate --

Compiling your contracts...
===========================
> Compiling .\contracts\Evote.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to D:\8th semester\fyp\development\final year project\e-voting-block-main\src\build\contracts
> Compiled successfully using:
   - solc: 0.8.12+commit.f00d7308.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0x98b539c146f55ee1956a6506f1cf208f1db82b98238701b6d1fb879f8f653a60
   > Blocks: 0            Seconds: 0
   > contract address:    0x92Da4A8b990d06283f04CBEE23EF9856F4e5F7F1
   > block number:        1
   > block timestamp:     1687846441
   > account:             0x1e5dBE58EfF8e50301831146E7cF416C96C4FdA4
   > balance:             99.99915577075
   > gas used:            250142 (0x3d11e)
   > gas price:           3.375 gwei
   > value sent:          0 ETH
   > total cost:          0.00084422925 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
```

Figure 8 Truffle Migration

1.2 MIGRATION SCREENSHOT:



```
C:\Windows\system32\cmd.exe
   -------------------------------------
   > Total cost:      0.00084422925 ETH


2_deploy_migration.js
=====================

   Deploying 'Evote'
   -----------------
   > transaction hash:    0x18f40b3e2a9aafec4bbc67b63a42de88685cf8d818797da5c22226a28abf198a
   > Blocks: 0            Seconds: 0
   > contract address:    0xcB45cFCe94312D7DE73a199d9529C9655c1Fe1fa
   > block number:        3
   > block timestamp:     1687846442
   > account:             0x1e5dBE58EfF8e50301831146E7cF416C96C4FdA4
   > balance:             99.994685974173391043
   > gas used:            1359028 (0x14bcb4)
   > gas price:           3.178365856 gwei
   > value sent:          0 ETH
   > total cost:          0.004319488192547968 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:     0.004319488192547968 ETH

Summary
=======
> Total deployments:   2
> Final cost:          0.005163717442547968 ETH


D:\8th semester\fyp\development\final year project\e-voting-block-main\src>
```

Figure 9 Truffle Migration 2

1. SECURITY:

1.1 IDENTIFICATION OF VULNERABILITY:

Vulnerability Found: When mathematical operations produce values outside of the maximum or minimum range permitted by the data type, unexpected behaviour and possible security hazards emerge.

Identified Functions: The addVote function, where candidatesList[id], is the contract's vulnerable function.The voteCount variable is increased using voteCount++.

Solution: To stop arithmetic overflows and underflows, use secure math libraries or apply manual checks.

1.2 REENTRANCY:

Reentrancy is the phenomenon of an external contract being able to frequently call back into the weak contract before the original execution is finished, causing unanticipated state changes and potential security breaches.

Identified Functions: The specified contract does not include the susceptible function. However, actions like transferring money that involve external contract calls and state changes must to be taken into account. Use mutex or reentrancy guard mechanisms, and implement the checks-effects-interactions pattern, where state changes are completed before any external calls.

1.3 ADDITIONAL SECURITY:

Input validation is an additional security measure.

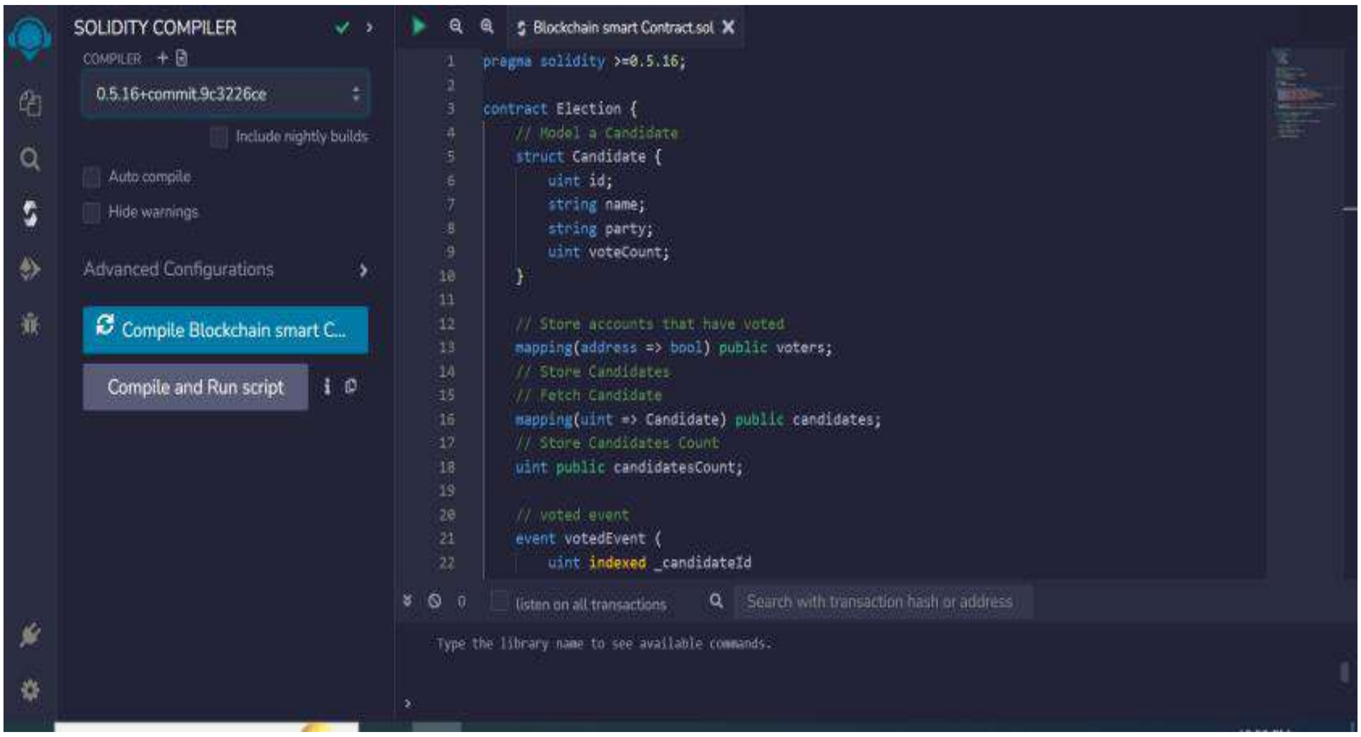
Identified Vulnerability: Inadequate input validation can result in a number of security problems, including data corruption, injection attacks, and unauthorized access.

Identified Functions: All commands such as createUser, createCandidate, etc. that take user input.

To guarantee that user-provided input is correctly vetted, sanitized, and satisfies the desired requirements, implement comprehensive input validation procedures. Securement techniques: Verify input parameters against pre-established norms, such as data kinds, length, format, or range limitations.

EVALUATION:

COMPILATION OF SMART CONTRACT:

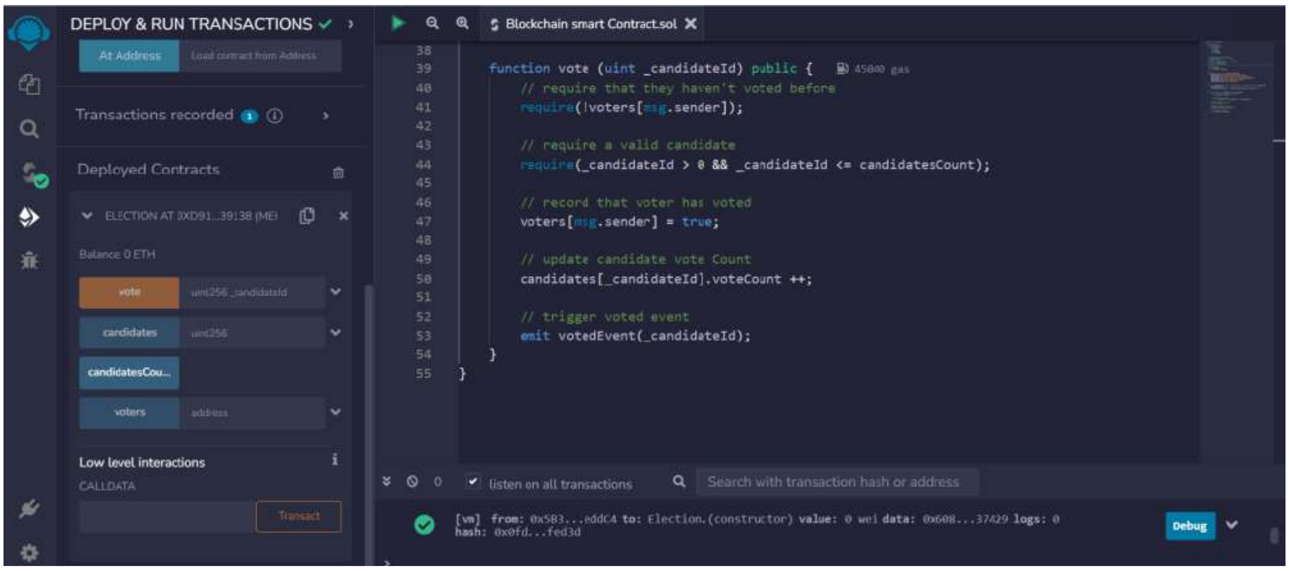


```
pragma solidity >=0.5.16;

contract Election {
    // Model a Candidate
    struct Candidate {
        uint id;
        string name;
        string party;
        uint voteCount;
    }

    // Store accounts that have voted
    mapping(address => bool) public voters;
    // Store Candidates
    // Fetch Candidate
    mapping(uint => Candidate) public candidates;
    // Store Candidates Count
    uint public candidatesCount;

    // voted event
    event votedEvent (
        uint indexed _candidateId
```

```
Type the library name to see available commands.
```

Figure 10 Smart Contract Compilation

DEPLOYMENT OF SMART CONTRACT:

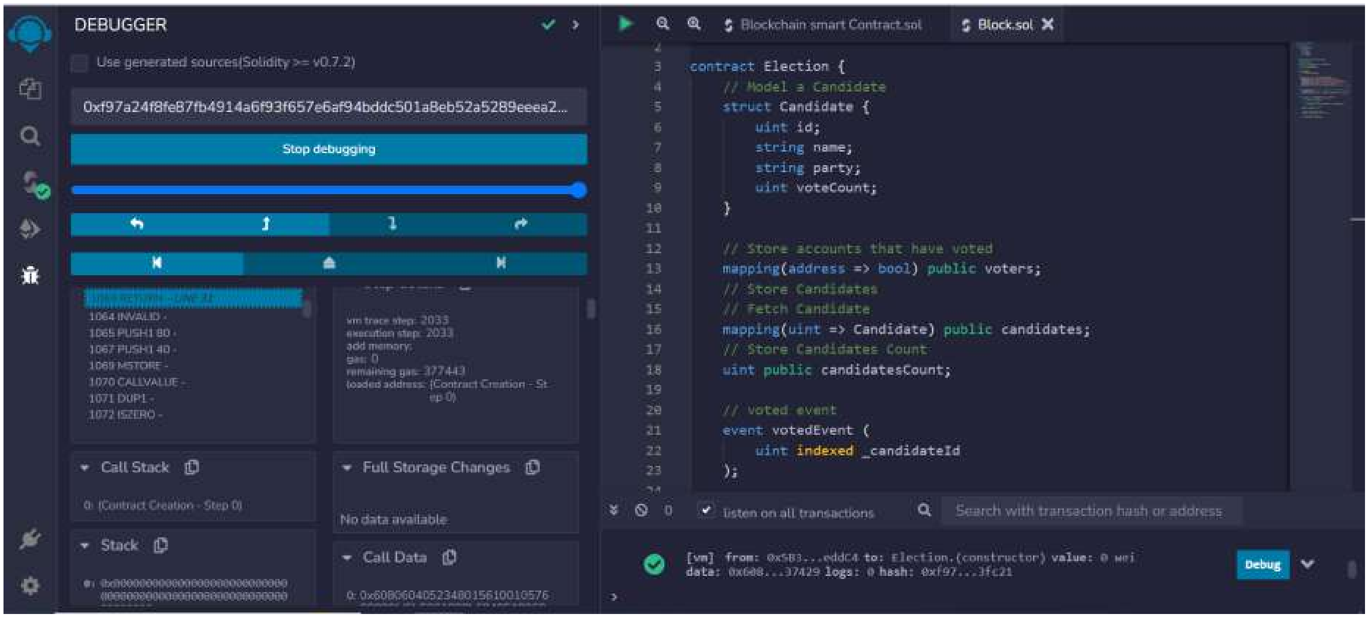


```
    function vote (uint _candidateId) public {    45040 gas
        // require that they haven't voted before
        require(!voters[msg.sender]);

        // require a valid candidate
        require(_candidateId > 0 && _candidateId <= candidatesCount);

        // record that voter has voted
        voters[msg.sender] = true;

        // update candidate vote Count
        candidates[_candidateId].voteCount ++;

        // trigger voted event
        emit votedEvent(_candidateId);
    }
}
```

```
[vm] from: 0x5B3...eddC4 to: Election.(constructor) value: 0 wei data: 0x608...37429 logs: 0
hash: 0x0fd...fed3d
```

Figure 11 Smart Contract Compilation 2

DEBUGGING:


DEBUGGER
Use generated sources(Solidity >= v0.7.2)
Stop debugging
Call Stack
Full Storage Changes
Stack
Call Data
Blockchain smart Contract.sol
Block.sol
listen on all transactions
Search with transaction hash or address
Debug

```
0xf97a24f8fe87fb4914a6f93f657e6af94bddc501a8eb52a5289eeea2...
```

```
1064 INVALID -
1065 PUSH1 80 -
1067 PUSH1 40 -
1069 MSTORE -
1070 CALLVALUE -
1071 DUP1 -
1072 ISZERO -
```

```
vm trace step: 2033
execution step: 2033
add memory:
gas: 0
remaining gas: 377443
loaded address: (Contract Creation - Step 0)
```

```
0: (Contract Creation - Step 0)
```

```
No data available
```

```
0: 0x000000000000000000000000000000000000000000000000000000000
```

```
0: 0x6080604052348015610010576
```

```
contract Election {
    // Model a Candidate
    struct Candidate {
        uint id;
        string name;
        string party;
        uint voteCount;
    }

    // Store accounts that have voted
    mapping(address => bool) public voters;
    // Store Candidates
    // Fetch Candidate
    mapping(uint => Candidate) public candidates;
    // Store Candidates Count
    uint public candidatesCount;

    // voted event
    event votedEvent (
        uint indexed _candidateId
    );
```

```
[vm] from: 0x5B3...eddC4 to: Election.(constructor) value: 0 wei
data: 0x608...37429 logs: 0 hash: 0xf97...3fc21
```

Figure 12 Debugging Smart Contract

FUTURE CONSIDERATION:

The blockchain-driven voting machine which would be involved in the future can be more developed by putting a significant weight on the support of integration, security, scalability, usability, verifiability, compliance, and education. Such security measures are necessitated in order to exclude unauthorized users in voting using the double voting system with the help of two-factor authentication. For the management of traffic volume that grows over time, increased utilization of sharding is imperative, as well as other scalability techniques. The visual appearance and quality of the product are also considered, next to making the user interface and instructions more comprehensible therefore increasing usability. Firstly, it is necessary to enhance the transparency of the process and make it easy to audit so that those interested can do that independently and their results will be reliable. Preserving the compliance with regulatory and legal responsibilities, which includes such as data privacy and elections, is critical. System acknowledgment and acceptance by the interval of education and promotion should be the ends of it. Amalgamation can take place because there will be a common policy like that, which is of voter registration and another one that takes care of result motivation.

CONCLUSION:

Such an approach of blockchain voting will pave way for the new era of voting transparency. From the technological point of view, the votes are cast without any error, and the fact that blockchain solves such problems due to it being immutable and decentralized. An automated voting is a kind of automation that reduces the probability of human mistakes and, besides, raises the accuracy of the procedure at the same time. Moreover, the elections can be more effective since the open connection of the blockchain itself can block the tyrant governor from making a new version to win. The last part of this section will be the

supervising of lies detection. In that case, the signer of the petition should be verified by the facial and electoral roll cross-reference systems, respectively. A smart contract is a piece of lists of scripts helping parties to determine the initial sequence of deployment and data. The truffle migrate command is used to activate all migration files and reset the status of migration. Truffle uses your Solidity smart contract code and uploads them to your desired Ethereum network. The code will be able to use deploy either multiple contracts or a single one with a migration script. The contract bytecode is stored on the blockchain and the transaction is created to deploy it.
Transaction fees: The Gas fees of Ethereum network need to be paid for deployment of contracts. Gas fees are to be incurred as an incentivization for a storage medium and computational process in the transaction of contracted deployment. It is suggested to fund your account with enough Ether (ETH) while going to public testnets or mainnet in particular.
Network configuration: In order to use Ethereum network as the preferred one you should set all the network parameters up in your truffle-config. js or truffle. js file. This comprises creating the parameters for node, which could be such as RPC endpoints, account numbers, and gas limits. Debugging and error handling: Truffle in the terminal will tell you if your contracts or migration scripts have any problems. Diagnosis and bug fixing are done with the help of error messages.

REFERENCES:

1. Garg, K., Saraswat, P., Bisht, S., Aggarwal, S.K., Kothuri, S.K. and Gupta, S., 2019, April. A comparitive analysis on e-voting system using blockchain. In *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)* (pp. 1-4). IEEE.
2. Fusco, F., Lunesu, M.I., Pani, F.E. and Pinna, A., 2018, September. Crypto-voting, a Blockchain based e-Voting System. In *KMIS* (pp. 221-225).
3. Vivek, S.K., Yashank, R.S., Prashanth, Y., Yashas, N. and Namratha, M., 2020, July. E-voting systems using blockchain: An exploratory literature survey. In *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)* (pp. 890-895). IEEE.
4. Hjálmarsson, F.Þ., Hreiðarsson, G.K., Hamdaqa, M. and Hjálmtýsson, G., 2018, July. Blockchain-based e-voting system. In *2018 IEEE 11th international conference on cloud computing (CLOUD)* (pp. 983-986). IEEE.
5. Noizat, P., 2015. Blockchain electronic vote. In *Handbook of digital currency* (pp. 453-461). Academic Press.
6. Hardwick, F.S., Gioulis, A., Akram, R.N. and Markantonakis, K., 2018, July. E-voting with blockchain: An e-voting protocol with decentralisation and voter privacy. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (pp. 1561-1567). IEEE.
7. Pawlak, M. and Poniszewska-Marańda, A., 2021. Trends in blockchain-based electronic voting systems. *Information Processing & Management*, *58*(4), p.102595.
8. Bogner, A., Chanson, M. and Meeuw, A., 2016, November. A decentralised sharing app running a smart contract on the ethereum blockchain. In *Proceedings of the 6th International Conference on the Internet of Things* (pp. 177-178).
9. Samreen, N.F. and Alalfi, M.H., 2020, February. Reentrancy vulnerability identification in ethereum smart contracts. In *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)* (pp. 22-29). IEEE.