

IERG 4998 IJ01

Final Year Project I

# Web application for e-voting using Blockchain and smart contract

By

Li, Chun Ngai

1155110647

A FINAL YEAR PROJECT REPORT  
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF BACHELOR OF INFORMATION ENGINEERING  
DEPARTMENT OF INFORMATION ENGINEERING  
THE CHINESE UNIVERSITY OF HONG KONG

APRIL 2021

## Abstract

The security of online voting has been questioned. Blockchain with smart contract is a good solution. In this project, we explore the advantages and disadvantages of Blockchain and its solutions by simulating a student poll. In order to make the voting more realistic and user-friendly, we put the voting function on the web application.

## 1. Introduction

There is no doubt that digitalization is the trend of today's society. Although digitalization brings a lot of convenience to people, their security has always been a hidden danger that should not be ignored. In the past, our solution was only to avoid carrying out things online that would have major security problems. However, due to the epidemic of the Corona Virus, we are forced to do things online that require maximum security, such as voting.

The benefit of online voting is that it saves time in counting votes and avoids invalid votes. However, an insecure voting system not only fails to ensure its fairness, but may also lead to a breach of voter information[1]. Blockchain and smart contracts, with their decentralized and unalterable properties, make them a perfect solution to ensure the fairness. The major usage of Blockchain is to record transactions for cryptocurrency, like Bitcoin. But nowadays people are trying to combine Blockchain with different areas due to its security. On the other hand, smart contracts handle and execute all transactions under this system. It guarantees that once the transaction has made, no one can modify or alter it. Which means that voting with Blockchain and smart contract retain the property of non-repudiation[2][3].

In this project, we focus on designing a smart contract for e-voting. To better represent what needs

to be taken into account when voting, we designed a scenario where students in a faculty vote for their favorite professor. We can also implement the diversity of voting in this way.

At the time we did this project, smart contracts and Blockchain were still highly discussed by the public. However, the public lacks the appropriate knowledge. The voting method thus cannot require the voters to have any knowledge on how to use Blockchain. Therefore, we are going to implement it in a web application so that it can be used more intuitively. Another benefit of making it a website is easy access, which is an important element in the design of an application for the general public.

## 2. Related Work

To implement a voting system. We have used **1. Solidity** to generate a smart contract. Then we use **2. Ganache** to launch a local server of Blockchain.

After that we have used **3. Truffle** to deploy a contract on Blockchain. There are two APIs that are usually used for interaction between web application, smart contract, and Blockchain. We chose **4. Drizzle** instead of **Web3** because Drizzle is developed with Truffle and Ganache.

Therefore, it is more compatible than Web3. For front end we use **5. React.js**, an open-source, JavaScript library. For user to interact with the transaction process, we used **6. MetaMask** to connect the address in Ganache.

User cannot vote until he receives the address and private key from Ganache's Blockchain. He can use those to connect to the web by using MetaMask.

### 3. Methodology

We first create a Solidity file to implement the smart contract.

```
/**
 * @dev Create a new ballot to choose one of 'proposalNames'.
 * @param proposalNames names of proposals
 */
constructor(uint[] memory proposalNames){
    chairperson = msg.sender;
    voters[chairperson].weight = 1;
    voters[chairperson].qualified = true;
    votingevent.totalproposal = proposalNames.length;
    votingevent.totalvote = 1;
    winningProposalPublic = 0;

    for (uint i = 0; i < proposalNames.length; i++) {
        // 'Proposal({...})' creates a temporary
        // Proposal object and 'proposals.push(...)'
        // appends it to the end of 'proposals'.
        proposals.push(Proposal({
            name: proposalNames[i],
            voteCount: 0
        }));
    }
}
```

```
/**
 * @dev Give 'voter' the right to vote on this ballot. May only be called by 'chairperson'.
 * @param voter address of voter
 */
function giveRightToVote(address voter) public {
    require(
        msg.sender == chairperson,
        "Only chairperson can give right to vote."
    );
    require(
        !voters[voter].qualified,
        "The voter already qualified."
    );
    voters[voter].weight = 1;
    candidate[votingevent.totalvote].Address = voter;
    votingevent.totalvote += 1;
    voters[voter].qualified = true;
}
```

```

/**
 * @dev Give your vote (including votes delegated to you) to proposal 'proposals[proposal].name'.
 * @param proposal index of proposal in the proposals array
 */
function vote(uint proposal) public {
    Voter storage sender = voters[msg.sender];
    require(sender.qualified, "Has no right to vote");
    //require(!sender.voted, "Already voted.");
    //sender.vote = proposal;

    // If 'proposal' is out of the range of the array,
    // this will throw automatically and revert all changes.
    if (!sender.voted){
        proposals[proposal].voteCount += 1;
        sender.weight = sender.weight - 1;
        if (sender.weight == 0)
            sender.voted = true;
    }
    if (sender.voted){
        proposals[proposal].voteCount += 0;
    }
}

```

```

/**
 * @dev Delegate your vote to the voter 'to'.
 * @param to address to which vote is delegated
 */
function delegate(address to) public {
    Voter storage sender = voters[msg.sender];
    require(!sender.voted, "You already voted.");
    require(to != msg.sender, "Self-delegation is disallowed.");
    require(voters[to].qualified, "You need to delegate to a qualified voter.");

    while (voters[to].delegate != address(0)) {
        to = voters[to].delegate;

        // We found a loop in the delegation, not allowed.
        require(to != msg.sender, "Found loop in delegation.");
    }
    sender.voted = true;
    sender.delegate = to;
    Voter storage delegate_ = voters[to];
    if (delegate_.voted) {
        delegate_.weight += sender.weight;
        delegate_.voted = false;
    } else {
        delegate_.weight += sender.weight;
    }
    /*
    if (delegate_.voted) {
        // If the delegate already voted,
        // directly add to the number of votes.
        proposals[delegate_.vote].voteCount += sender.weight;
    } else {
        // If the delegate did not vote yet,
        // add to her weight.
        delegate_.weight += sender.weight;
    }
    */
}

```

```

/**
 * @dev Computes the winning proposal taking all previous votes into account.
 * @return winningProposal_ index of winning proposal in the proposals array
 */
//bool public tieCheck = false;
function winningProposal() public view
|   returns (uint winningProposal_)
{
    uint winningvoteCount = 0;
    //bool tieCheck = false;
    for (uint p = 0; p < proposals.length; p++) {
        if (proposals[p].voteCount > winningvoteCount) {
            winningvoteCount = proposals[p].voteCount;
            winningProposal_ = p;
            //tieCheck = false;
        } else if (proposals[p].voteCount == winningvoteCount) {
            //tieCheck = true;
        }
    }
    // If 'tieCheck' is true,
    // which mean there are more than one winner,
    // it will not return anythings.
    // you can use viewRank() to check the result.
    //require(tieCheck == false, "The event got tiebreak.");
}

```

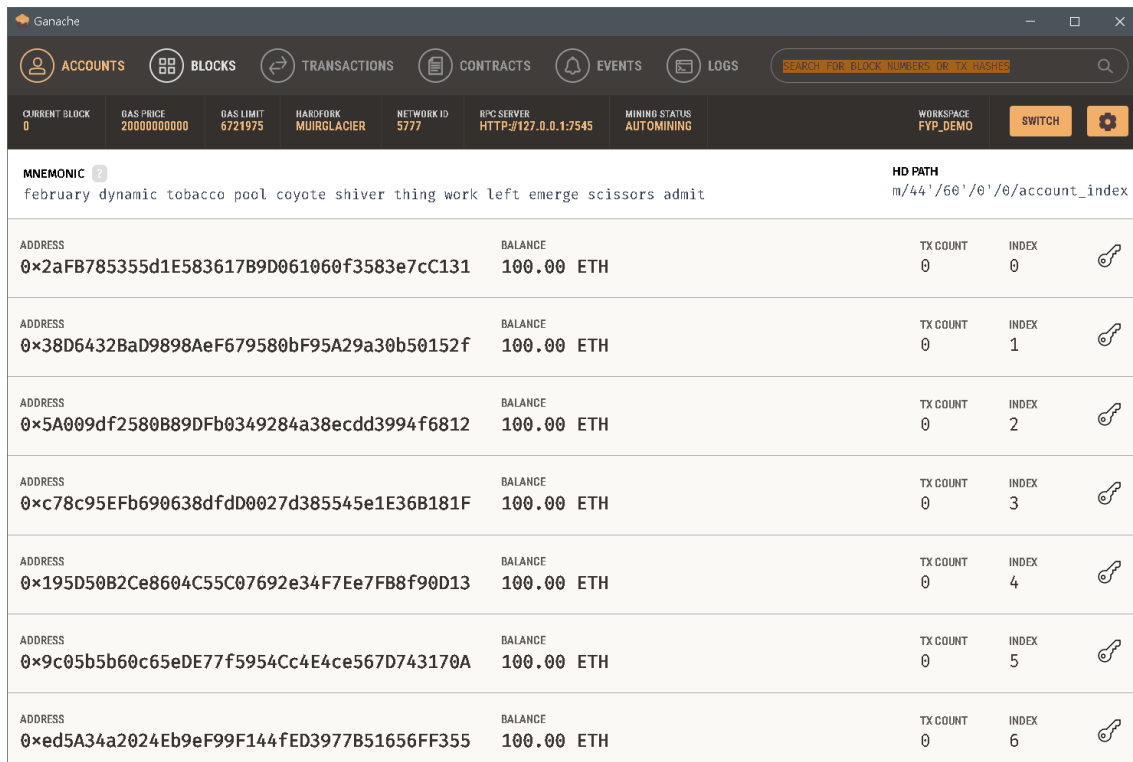
```

/**
 * @dev Call viewRank() function to get the rank of the proposal
 * @return rankName ranked proposal name
 * @return rankVote ranked proposal vote count
 */
function viewRank() public view
|   returns (uint[] memory rankName, uint[] memory rankVote)
{
    Proposal[] memory tempProposal = new Proposal[](votingevent.totalproposal);
    for (uint i = 0; i < votingevent.totalproposal; i++){
        tempProposal = proposals;
    }
    Proposal memory swapProposal;
    for (uint k = 0; k < votingevent.totalproposal; k++){
        for (uint j = 0; j < votingevent.totalproposal - 1 - k; j++){
            if (tempProposal[j].voteCount < tempProposal[j + 1].voteCount){
                swapProposal = tempProposal[j];
                tempProposal[j] = tempProposal[j + 1];
                tempProposal[j + 1] = swapProposal;
            }
        }
    }
    uint[] memory tempname = new uint[](votingevent.totalproposal);
    for (uint p = 0; p < votingevent.totalproposal; p++){
        tempname[p] = tempProposal[p].name;
    }
    uint[] memory tempvote = new uint[](votingevent.totalproposal);
    for (uint q = 0; q < votingevent.totalproposal; q++){
        tempvote[q] = tempProposal[q].voteCount;
    }
    return (tempname, tempvote);
}

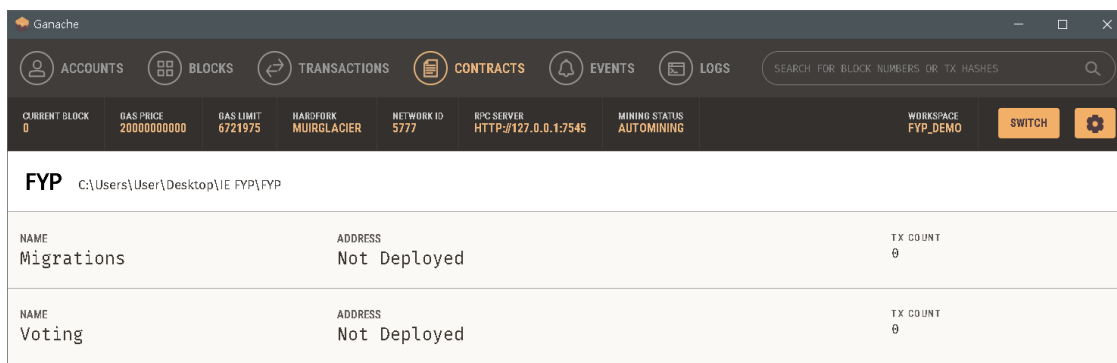
```

Basically it includes all the function we need in this voting session. In our example, we have the chairperson (admin) to give vote to the students (voters). Students can also delegate their votes to the others. And it will determines who is the winner whenever someone has voted, and provide the rank of every candidate.

Then we set up a Ganache workspace for a local Blockchain server.

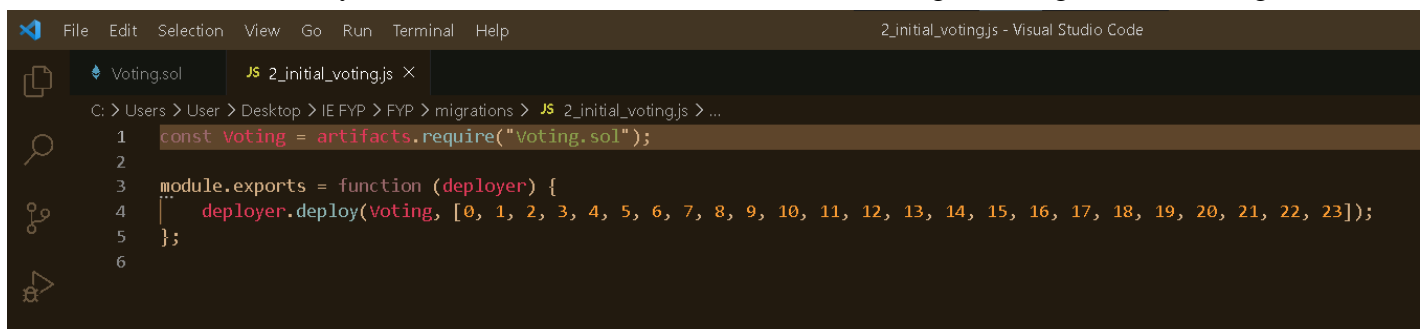


ADDRESS	BALANCE	TX COUNT	INDEX
0x2aFB785355d1E583617B9D061060f3583e7cC131	100.00 ETH	0	0
0x38D6432BaD9898AeF679580bF95A29a30b50152f	100.00 ETH	0	1
0x5A009df2580B89DFb0349284a38ecdd3994f6812	100.00 ETH	0	2
0xc78c95EFb690638dfd0027d385545e1E36B181F	100.00 ETH	0	3
0x195D50B2Ce8604C55C07692e34F7Ee7FB8f90D13	100.00 ETH	0	4
0x9c05b5b60c65eDE77f5954Cc4E4ce567D743170A	100.00 ETH	0	5
0xed5A34a2024Eb9eF99F144fED3977B51656FF355	100.00 ETH	0	6



NAME	ADDRESS	TX COUNT
Migrations	Not Deployed	0
Voting	Not Deployed	0

We must decide how many candidates in advance, and it cannot be changed throughout this voting session.



```
1 const voting = artifacts.require("Voting.sol");
2
3 module.exports = function (deployer) {
4   deployer.deploy(voting, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]);
5 };
6
```

After that, you may deploy the Truffle onto the Blockchain.

```
CAVINDOWS\system32\cmd.exe
2_initial_voting.js

Replacing 'Voting'
-----
> transaction hash: 0x377e091181489bd721f0aa1732281c7309e71f7c16ecde15910e5dca2c308b6f
> Blocks: 0
> contract address: 0x7eF748003ae217Aeb77bB20E2C652dAa64577942
> block number: 3
> block timestamp: 1619518835
> account: 0x2aFB785355d1E583617B9D061060f3583e7cC131
> balance: 99.95875713
> gas used: 1832843 (0x1bf78b)
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.03665686 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0.03665686 ETH

Summary
> Total deployments: 2
> Final cost: 0.04039612 ETH
```

One can always check the status of the contract in Ganache.

Ganache

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK4

GAS PRICE2000000000

GAS LIMIT6721975

HARDFORKMUIRGLACIER

NETWORK ID5777

RPC SERVERHTTP://127.0.0.1:7545

MINING STATUSAUTOMINING

WORKSPACEFYP\_DEMO

SWITCH


FYP C:\Users\User\Desktop\FYP\FYP

NAME	ADDRESS	TX COUNT	DEPLOYED
Migrations	0x5683013e6a4d3beE0c799b987594515b2830A90a	1	
Voting	0x7eF748003ae217Aeb77bB20E2C652dAa64577942	0	

Then we created a website interface using React.js.

ReactApp

localhost:3000



# Welcome to the Blockchain Demo Voting Session!

## Choose your favourite Lecturer!

View Candidates

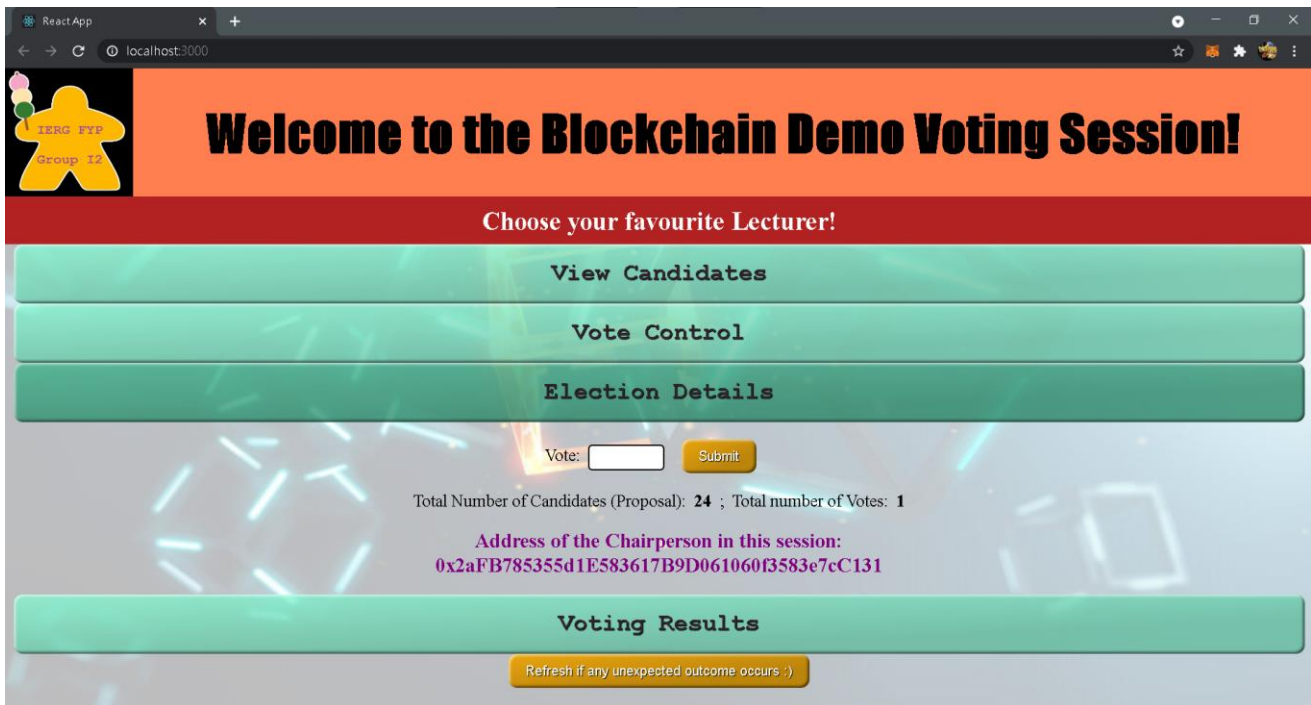
Vote Control

Election Details

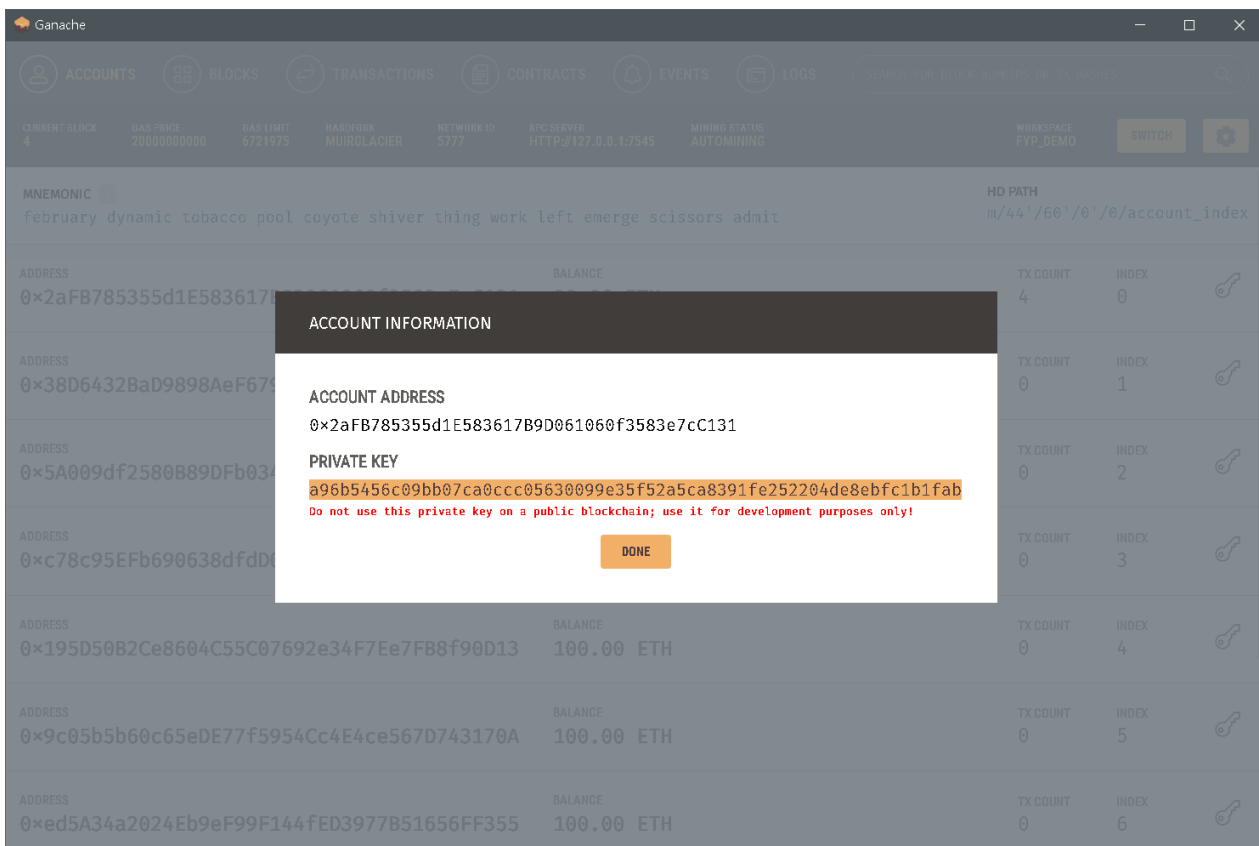
Voting Results

Refresh if any unexpected outcome occurs :)

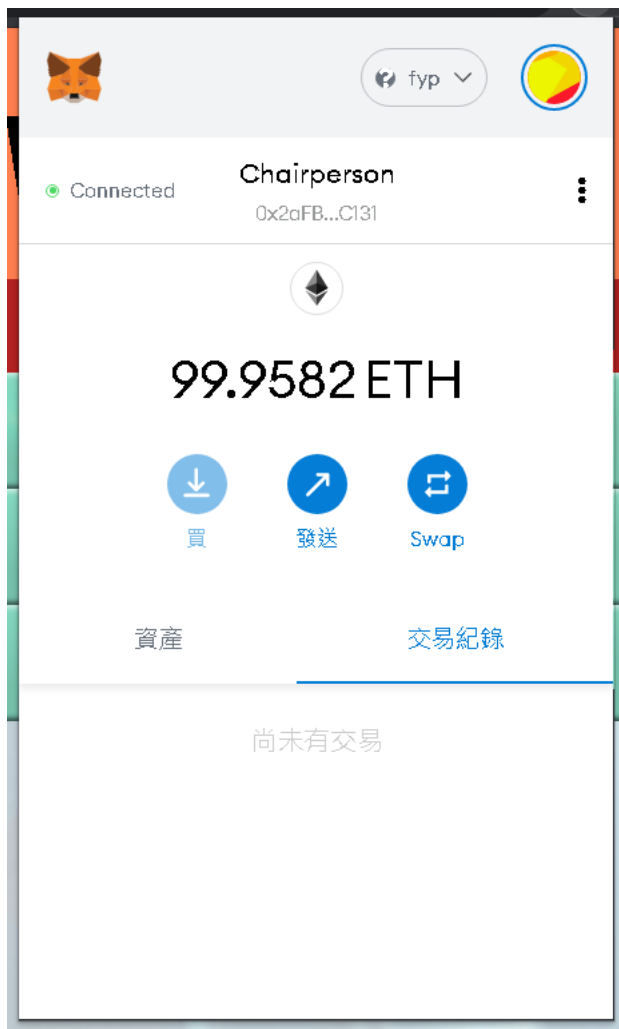
In the “Election Details”, one can check the address of the chairperson, number of proposals (candidates), total number of the vote (voted and not voted are both counted).



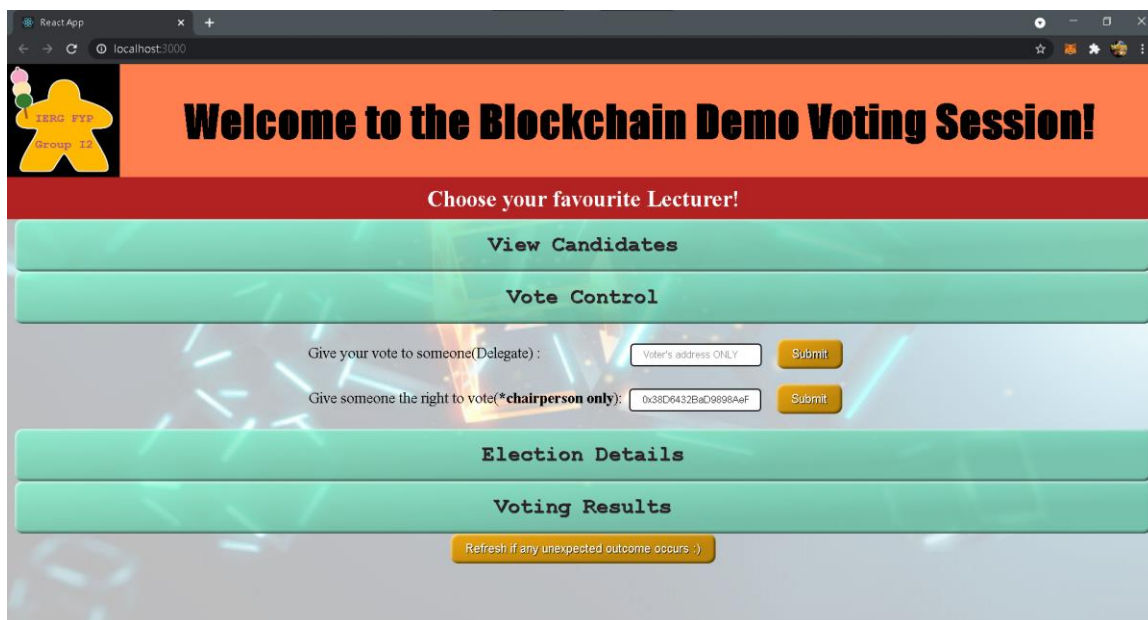
In the Ganache, the admin can distribute the private to the voter for to use in MetaMask.



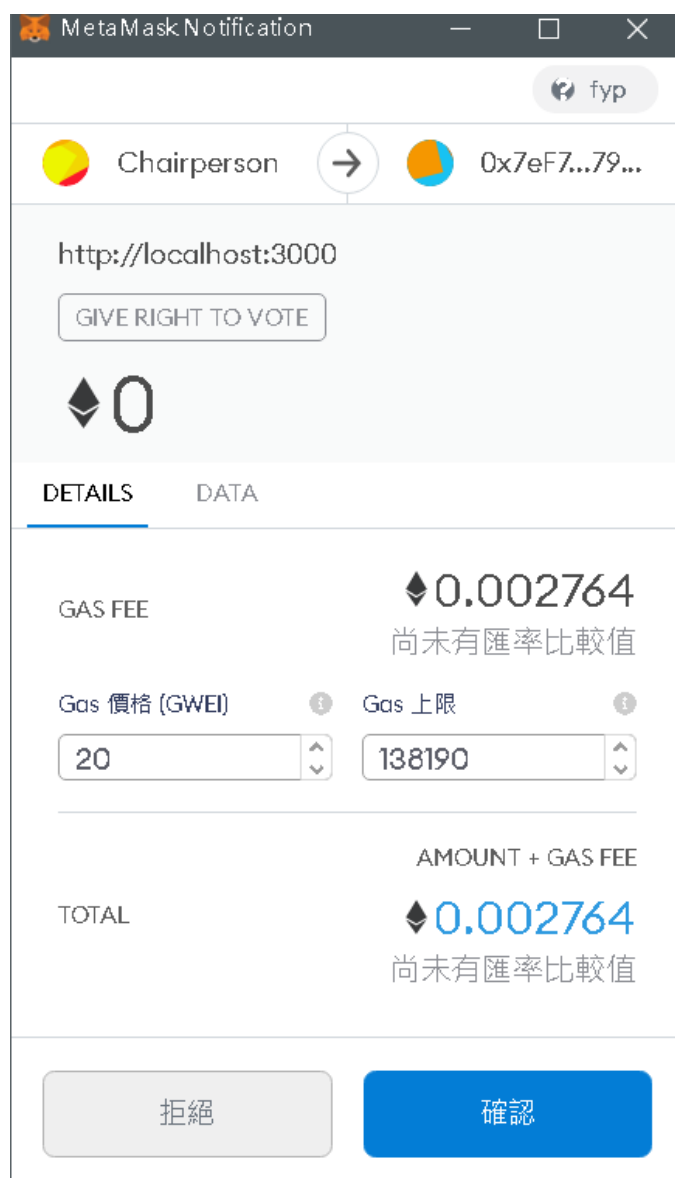




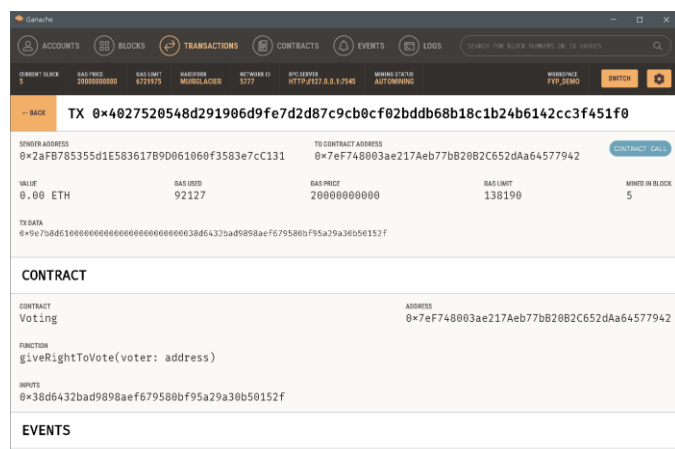
In “Vote Control”, admin can give someone the right to vote, while normal voters can delegate their vote here.



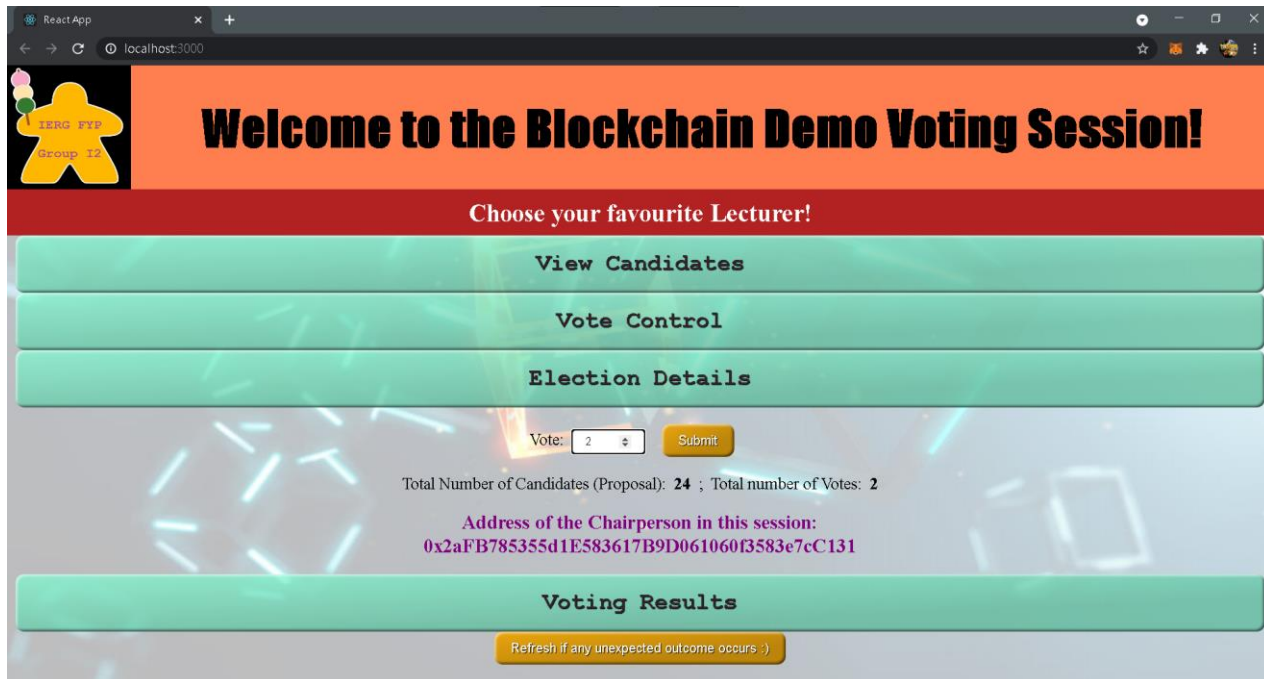
Every process required Gas Fee, including chairperson give right to someone, and voters' delegation. Paying zero Gas Fee is also allowed as it is a virtual local Blockchain.



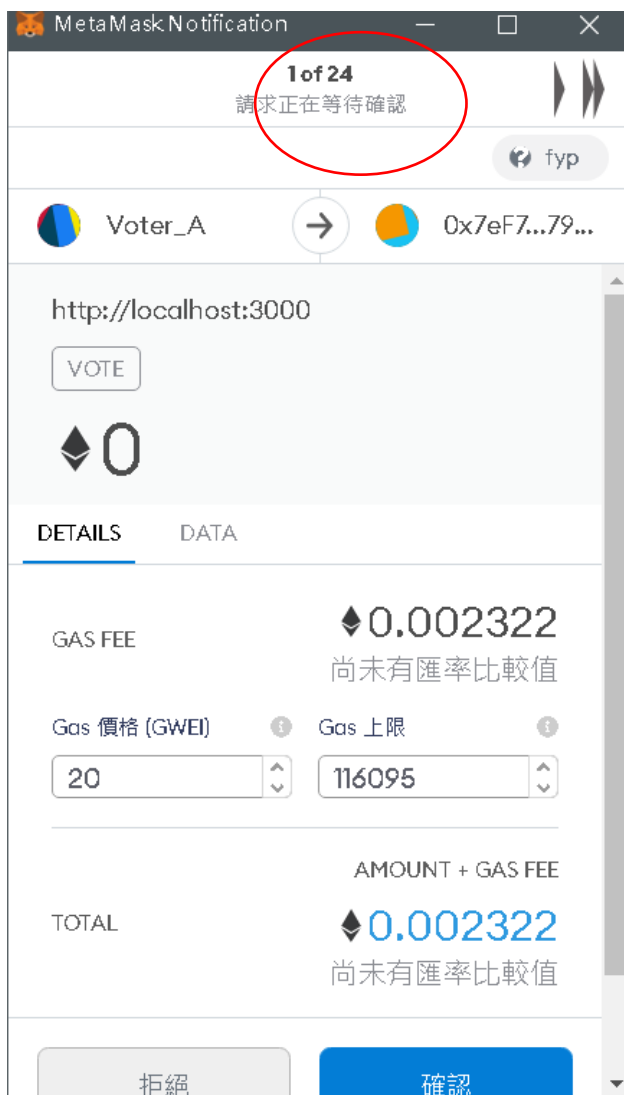
In Ganache, everyone can see every transaction created. Which provide perfect integrity.



In “Election details”, one can vote by choosing a candidate. If one has two votes (being delegated), he has to vote separately.

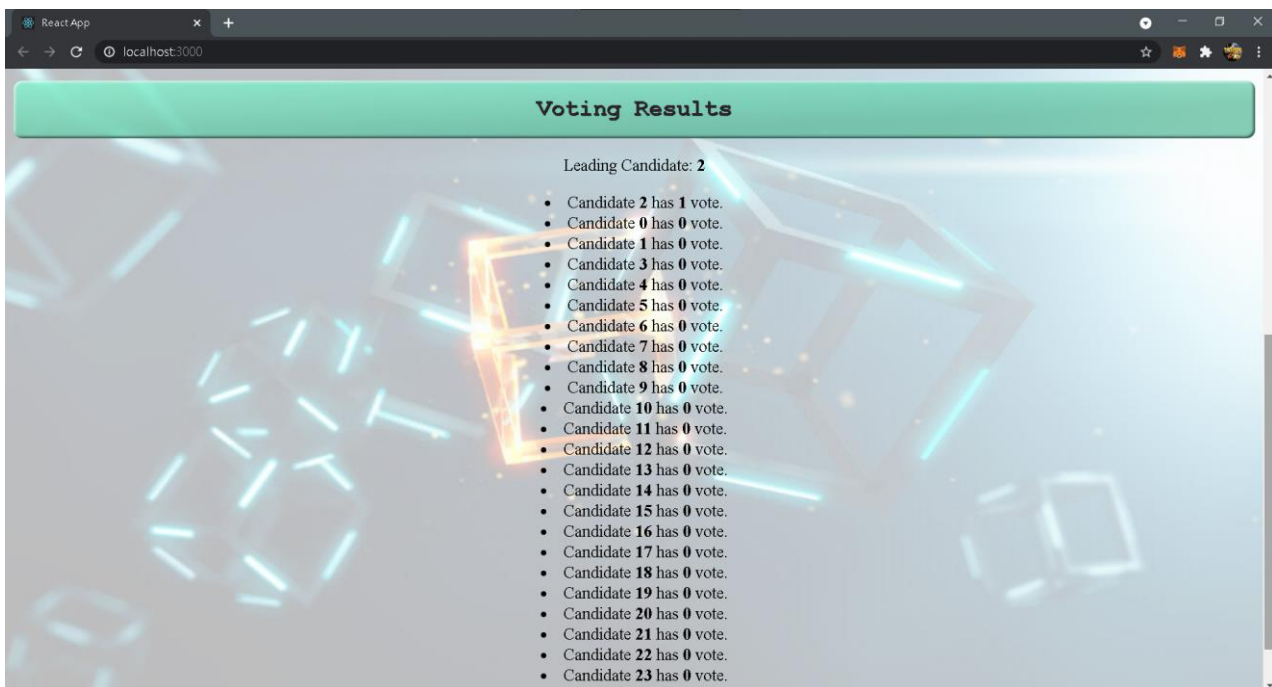


In order to provide secrecy, other candidates will also be voted with a dummy(weightless) vote. Thus a receipt is also created in MetaMask and Ganache.

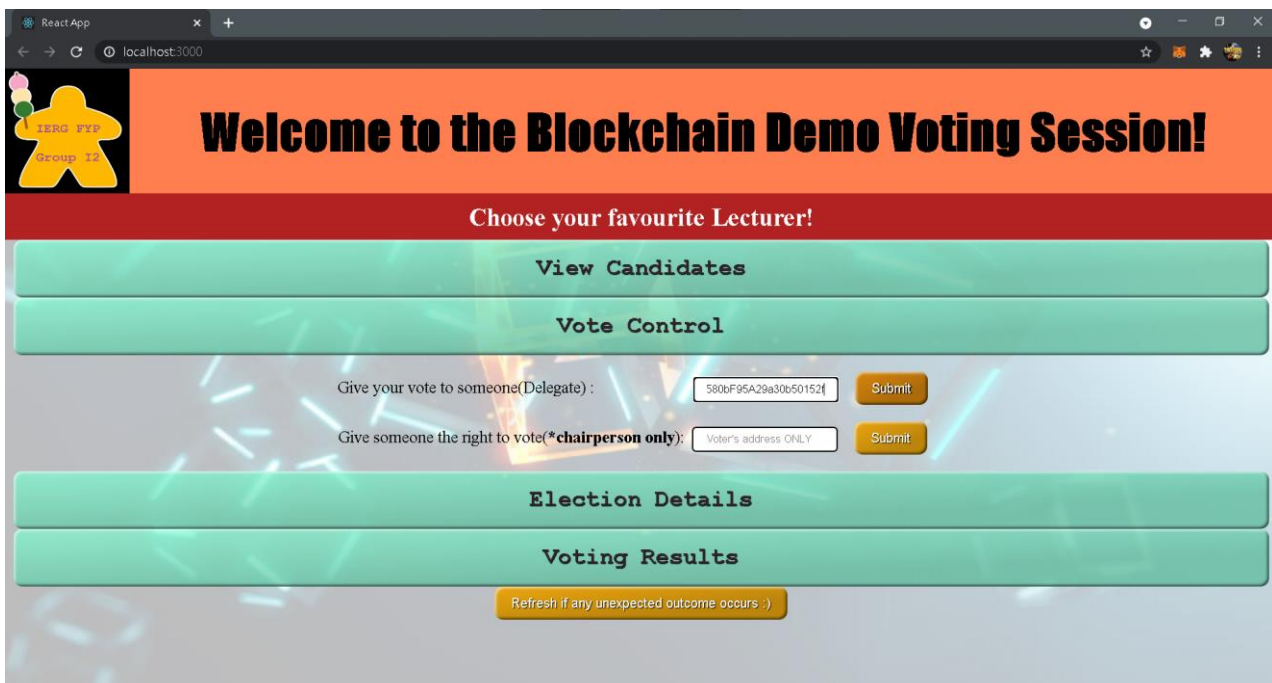


ADDRESS	BALANCE	TX COUNT	INDEX
0x38D6432BaD9898AeF679580bF95A29a30b50152f	99.99 ETH	24	1

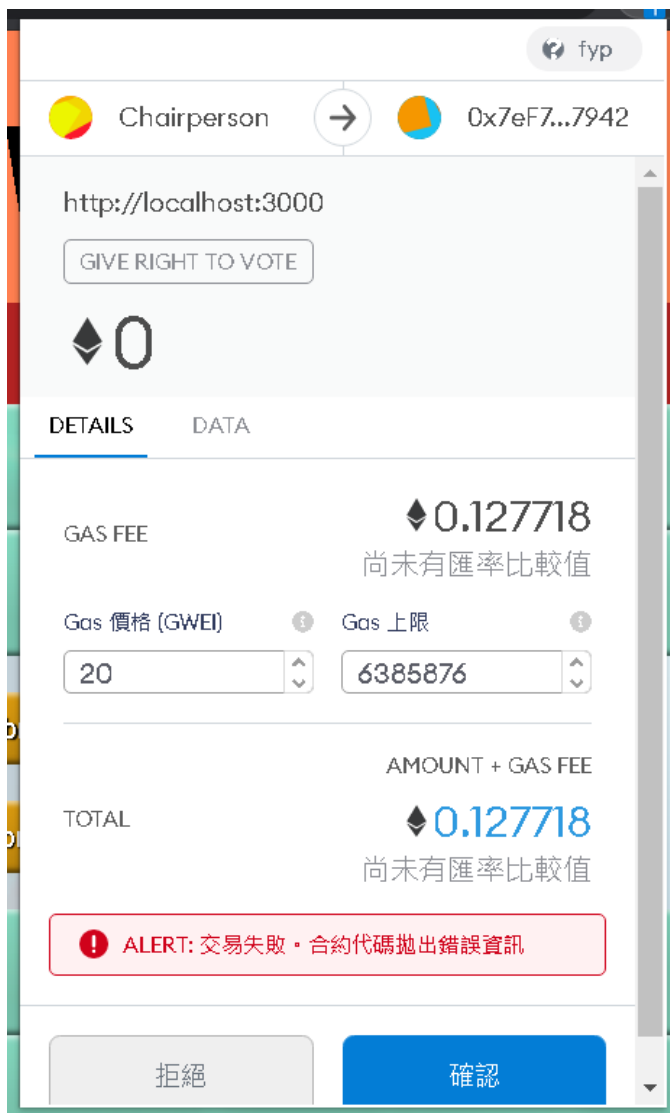
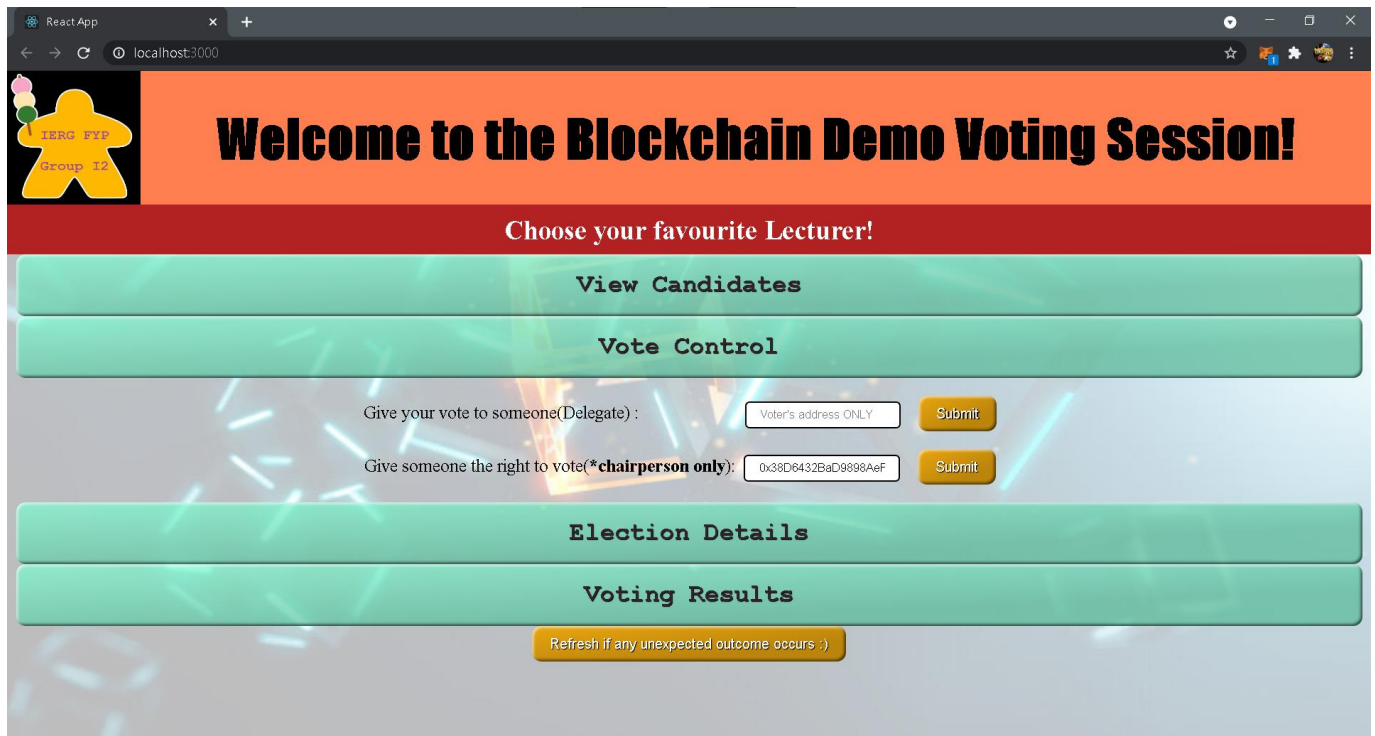
In “Voting Results”, one may always check the instant result of the voting session.



When using delegate function, one may delegate his vote to anyone who is eligible to vote, even though the voters has already used his vote. It is also traceable in Ganache.







## 4. Result

This web application is able to perform proper voting session. Every voting will generate receipt.

For delegation, many marginal cases are avoided, for example one cannot delegate vote to itself, and delegation that causes a loop is forbidden (i.e., A delegate B; B delegate C; and C delegate A).

For the transaction fee. Since it is a local Blockchain, there is not minimum price for every action. If a voting session in real life has cost, this implementation may not be the best as it will generate a lot of dummy votes to mask users' choices, causing unwanted expenses.

As a result, we think that Blockchain provide adequate integrity. However, confidentiality cannot be ensured if one use the Blockchain and Smart Contract only. More tools should be used together to provide better security.

## 5. Conclusion

In conclusion, Blockchain and smart contract works perfectly for the open ballot. Since it can ensure fairness and integrity in voting. However, every transaction is recorded in the Blockchain and every user are able to access the chain. It indicates that if one's address is known by someone, his choice is also exposed. In order to minimize the chance of exposure, whenever a voter make his choice, the system will automatically create dummy transactions for all the candidates he does not choose. Therefore users cannot know who he votes for.

However, this method is still imperfect. For example, users may be able to guess the pattern of the dummy transactions. On the other hand, if the organization choose to use an admin to give voters the right to vote, the admin will know every one's addresses and it will be very dangerous. These misinterpretations should be aware and carefully consider when using Blockchain as a voting system.

Another disadvantage is the compiling time and compiling power is highly demanded. Since we have to mask the result, dummy transactions are generated. A normal voting should receive at most  $N$  votes, where  $N$  is the number of voters. However, our implementation will receive  $N*M$  votes, where  $M$  is the number of Candidate. The computing time will thus increase from  $O(n)$  to  $O(n^2)$ . This is also a key point to note, whether it is worthy or not to give up transaction speed to enhance limited confidentiality.

## 6. Further Discussions

To further optimize the confidentiality, first we can abandon the admin mechanism. By using some open source Blockchain, we can create our own chain without using the address provided by Ganache or Ethereum mainnet. However, that may create another problem if we do not control who can sign up for our own chain. Back to our students voting example. By whitelisting certain domain (i.e., *link.cuhk.edu.hk* or *ie.cuhk.edu.hk*), we may ensure only CUHK students or IE major students are eligible to vote.

Another solution to prevent cheating by admin is using encryption like End-to-end encryption (E2EE). For example WhatsApp has used this to promise that their company have no access to users' messages. E2EE can ensure admin do not know any address-email pair after giving right to voters. But this implementation is already beyond the usage of Blockchain.

## 7. Appendix: source code of voting system

## 8. Reference

F. Mouton, M. M. Malan, L. Leenen and H. S. Venter, "Social engineering attack



framework," 2014 Information Security for South Africa, Johannesburg, 2014, pp. 1-9, doi: 10.1109/ISSA.2014.6950510.  
<https://ieeexplore-ieee-org.easyaccess1.lib.cuhk.edu.hk/document/6950510>

Protections. IEEE Access, 8, 24416-24427.  
[https://julac.hosted.exlibrisgroup.com/permalink/f/1h1uk14/TN\\_cdi\\_crossref\\_primary\\_10\\_1109\\_ACCESS\\_2020\\_2970495](https://julac.hosted.exlibrisgroup.com/permalink/f/1h1uk14/TN_cdi_crossref_primary_10_1109_ACCESS_2020_2970495)

Daramola, Olawande, & Thebus, Darren. (2020). Architecture-Centric Evaluation of Blockchain-Based Smart Contract E-Voting for National Elections. Informatics (Basel), 7(2), 16.  
[https://julac.hosted.exlibrisgroup.com/permalink/f/1h1uk14/TN\\_cdi\\_doaj\\_primary\\_oai\\_doaj\\_org\\_article\\_49bd0ab1bfd44376b2362f5d9e7e3bbd](https://julac.hosted.exlibrisgroup.com/permalink/f/1h1uk14/TN_cdi_doaj_primary_oai_doaj_org_article_49bd0ab1bfd44376b2362f5d9e7e3bbd)

Tso, Raylin, Liu, Zi-Yuan, & Hsiao, Jen-Ho. (2019). Distributed E-Voting and E-Bidding Systems Based on Smart Contract. Electronics (Basel), 8(4), 422.  
[https://julac.hosted.exlibrisgroup.com/permalink/f/1h1uk14/TN\\_cdi\\_doaj\\_primary\\_oai\\_doaj\\_org\\_article\\_07d36cd2869d4c778570d778eb25a254](https://julac.hosted.exlibrisgroup.com/permalink/f/1h1uk14/TN_cdi_doaj_primary_oai_doaj_org_article_07d36cd2869d4c778570d778eb25a254)

“Solidity by Example”, Solidity  
<https://solidity.readthedocs.io/en/v0.7.0/solidity-by-example.html>

“Statement Voting”, Bingsheng Zhang and Hong-Sheng Zhou, 2017  
<https://eprint.iacr.org/2017/616>

Atapattu, Charith, & Chung, Sam. (2018). An Architectural Analysis of a Blockchain-Based Web Application for Supporting Smart Contracts and Hardening Security. 186.  
[https://julac.hosted.exlibrisgroup.com/permalink/f/1h1uk14/TN\\_cdi\\_acm\\_primary\\_3241827](https://julac.hosted.exlibrisgroup.com/permalink/f/1h1uk14/TN_cdi_acm_primary_3241827)

Sayeed, Sarwar, Marco-Gisbert, Hector, & Caira, Tom. (2020). Smart Contract: Attacks and