Blockchain Global Voting System
EECE 571G Final Project Report

Xiaojun Tian 11240587

Liming Liu 95660163

Yuwei Luo 14152342

Jiahao Li 50462670

April 16, 2020

Contents

1 Introduction

From government change to student union election, voting is very common in daily life. Since the day voting was invented, it has been the most intuitive form to show the will and proposition of voters. However, the shortcoming of the traditional election model will cause more problems, which will deviate it from its original purpose.

Vote can often be inconvenient, time-consuming, and most importantly, unsafe. For example, the polling day of the US Presidential Election is set on Tuesday and people have to leave work, waiting in long lines before conducting a vote. Besides, the vote is immutable so people do not have the chance to alter their selection. Online voting does solve the inconvenient and time-consuming problems, however, new problems also arise such as account stealing and malicious attack towards the server or database.

Furthermore, the authenticity of the voting results is a big concern since either in-person voting or online voting has only one central authority. It is easy to modify or manipulate the data from the interior. In 2016, the voting results show that The Greens party of Austria has a 146.9% support level in one city, which apparently means that a large number of votes are actually forged and the whole election system is manipulated.

These problems are the main motivations of why we would like to apply blockchain technologies on the voting system. First, blockchain is decentralized with a network consisting of hundreds and thousands of machines, rather than a centralized server. Therefore, if a hacker is trying to block the system, he/she will have to take down all the distributed hosing nodes, which is almost impossible. As a result, we can significantly decrease the risk of being attacked, and guarantee the authenticity of the voting results. On the other hand, since the data stored in the network is allowed to be distributed across all its nodes, nodes will not affect each other and they run on the network accordingly. Thus the blockchain based D-vote-app is more fault-tolerant. Besides, since all the blocks are chained using cryptography, the transactions will be added immutably and preserved permanently for all to check, hence the stored data is transparent and unalterable.

2 Use Case Diagrams

In this project, we mainly focus on implementing the two most common voting processes, straight voting and cumulative voting. In a straight voting, for example, you have 20 shares so each candidate can have up to 20 votes from you according to your preference. In a cumulative voting, for example, you have 80 shares and you can allocate these shares among all the candidates in whichever way you want. In the real world, straight voting is more common but cumulative voting allows minority shareholders to have a greater chance of influencing who gets elected to the board.

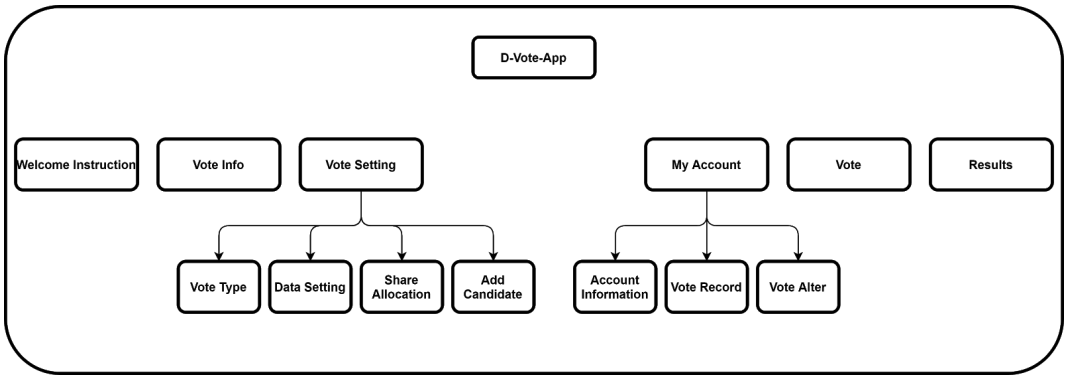

Figure 1: Application level

Our D-vote-app has 6 main modules as Figure 1 shows, including welcome instruction, vote info, vote setting, my account, vote, and results. Besides, there are 3 and 4 sub-modules under the vote setting module and my account module, respectively.

The application is orientated to two types of users, deployers and voters. First, the deployer deploys the contract and complete vote setting such as vote type, vote period etc. Then, the valid voter can vote for their favorite candidates and check the real-time results during or after the voting period. The following flow chart Figure 2 shows the whole process for our D-vote-app.

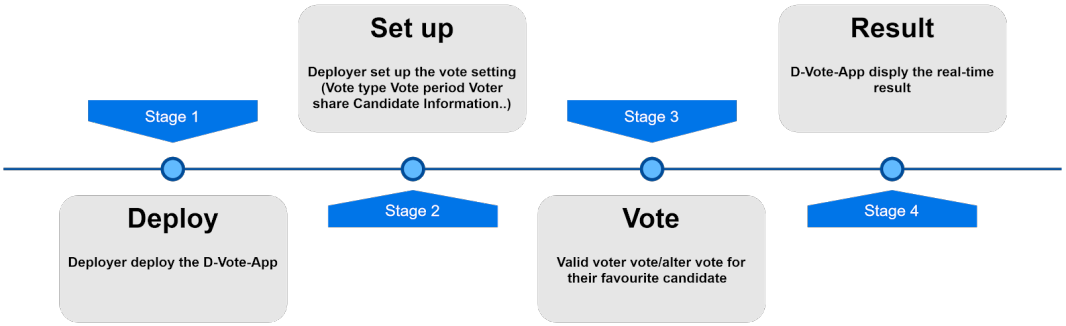

Figure 2: Application flow chart

2.1 Deployer

The deployer creates a poll by deploying the smart contract and then becomes the 'Administrator' in the voting system. The basic authority of the deployer is defining voting properties, including voting type (statutory or cumulative), voting period, outstanding shares, number of open seats, and the maximum amount of shares per voter can obtain. Besides, the deployer can also add candidate information and allocate shares among voters. Figure 3 shows the summary of a deployer's authorities and Figure 4, Figure 5, Figure 6 show the corresponding details.

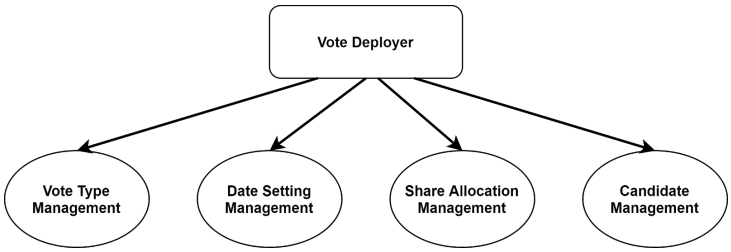

Figure 3: Authorities of a deployer

2.1.1 Date Setting

Figure 4: Date setting

Our D-vote-app allows the deployer to set vote type and valid vote period in the front-end GUI. Other vote settings such as outstanding shares, open seats, and the maximum

amount of shares per voter need to be determined during the contract deploying period.

2.1.2 Share Allocation

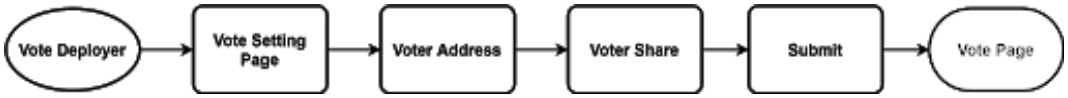

Figure 5: Share allocation

After the deployer has set the vote type and period, the deployer can navigate to the vote setting page to allocate shares among voters by inputting the voter's address and amount of shares.

2.1.3 Candidate Setting

Figure 6: Candidate setting

For the candidate information, the deployer can input data through vote setting page and specify the candidate name and photo as well as a short description.

2.2 Voter

After the poll has been created, the voter can view the candidates information and vote for candidates as long as he/she has enough shares during the valid voting period. Then, the voter can view personal voting records, personal available shares, and the real-time voting results through GUI. Each voter has 3 chances to modify the vote. Once the voter has changed a vote, the voting results, voting records, and personal available shares are refreshed. Figure 7 shows the summary of a deployer's authorities and Figure 8, Figure 9, Figure 10, Figure 11, Figure 12 show the corresponding details.

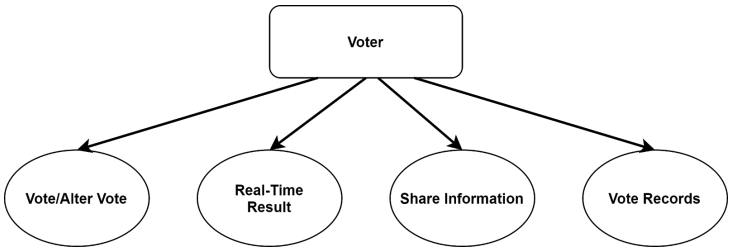

Figure 7: Authorities of a voter

2.2.1 Vote Process

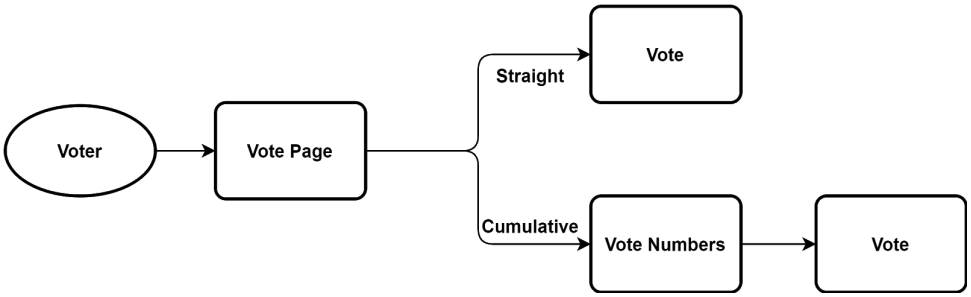

Figure 8: Vote Process

For the vote process, the voter can directly vote if the vote type is straight. The voter needs to specify the vote numbers if the vote type is cumulative.

2.2.2 Alter Vote Process

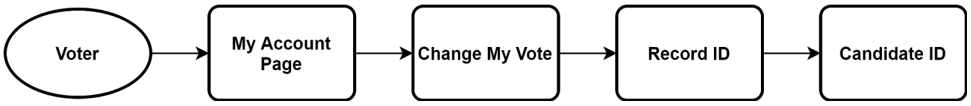

Figure 9: Alter vote process

For the alter vote process, the voter can input data through my account page and specify the record ID and candidate ID.

2.2.3 Vote Records

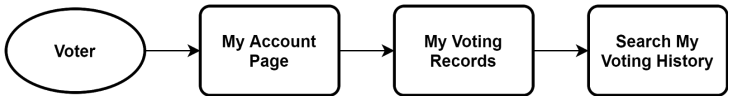

Figure 10: Vote records

For the vote records, the voter can navigate to my account page to view the voting records.

2.2.4 Share Information

Figure 11: Share information

For the share information, the voter can navigate to my account page to view their personal share information.

2.2.5 Real-time Result

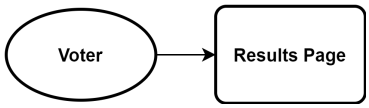

Figure 12: Real-time result

For the real-time result, the voter can navigate to results page to view the real-time and final voting result.

2.3 Summary of User Cases

Figure 13 summaries all the user cases as mentioned above.

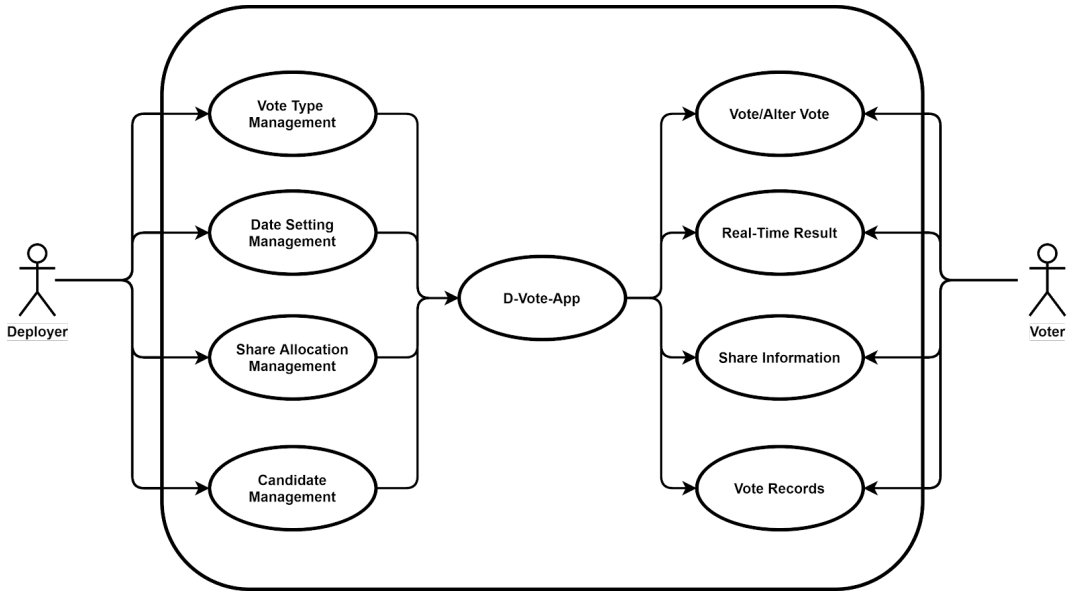

Figure 13: Summary of the user cases

3 Instructions for Use

3.1 Configuration Settings

```
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "bootstrap": "^4.4.1",
    "moment": "^2.24.0",
    "react": "16.11.0",
    "react-bootstrap": "^1.0.0-beta.17",
    "react-date-picker": "^7.10.0",
    "react-dom": "16.11.0",
    "react-google-charts": "^3.0.15",
    "react-router-dom": "^5.1.2",
    "react-scripts": "3.2.0",
    "semantic-ui-react": "^0.88.2",
    "web3": "1.2.2"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": "react-app"
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "devDependencies": {}
}
```

Figure 14: npm configuration

```
const path = require("path");
module.exports = {
  contracts_directory: './contracts/',
  contracts_build_directory: path.join(__dirname, './client/src/contracts'),
  networks: {
    development: {
      host: "127.0.0.1",
      port: 7545,
      network_id: "*"
    }
  }
};
```

Figure 15: Truffle configuration

```
xiaojuntian@Xiaojuns-MBP EECE571G_BLOCKCHAIN % truffle compile

Compiling your contracts...
===========================
> Compiling ./contracts/Vote.sol
> Artifacts written to /Users/xiaojuntian/UBC/EECE571G_BLOCKCHAIN/client/src/contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
```

Figure 16: Contract compile

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


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

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0x6ec5332bee62f911d71780b3f1a9ae6d3e26bae7eb200da667a7d1a366df99b3
   > Blocks: 0            Seconds: 0
   > contract address:    0x57f42A574194cDfDb1b4528E4213f204F90488A5
   > block number:        745
   > block timestamp:     1586380449
   > account:             0xDE7bFC5fC14665F5A6725E4070E1030Sbf475f78
   > balance:             95.11075294
   > gas used:            188483
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00376966 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00376966 ETH


2_deploy_contracts.js
=====================

   Replacing 'Vote'
   ----------------
   > transaction hash:    0x43bc900197858a2a0192b38daae3528a63bdf41883012822de726732a34ed423
   > Blocks: 0            Seconds: 0
   > contract address:    0x0aD7dA4A97d3bc7485f6b056dBcA19C652b281f2
   > block number:        747
   > block timestamp:     1586380450
   > account:             0xDE7bFC5fC14665F5A6725E4070E1030Sbf475f78
   > balance:             95.00577736
   > gas used:            5606770
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.11213556 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.11213556 ETH


Summary
=======
> Total deployments:   2
> Final cost:          0.11590522 ETH
```

Figure 17: Contract migration

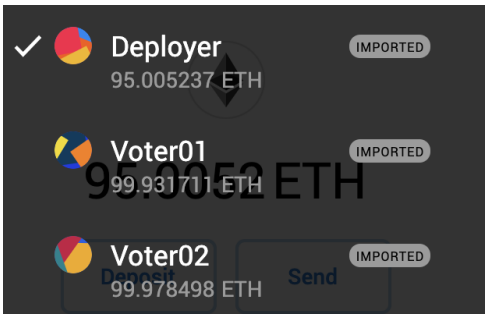

Figure 18: MetaMask configuration

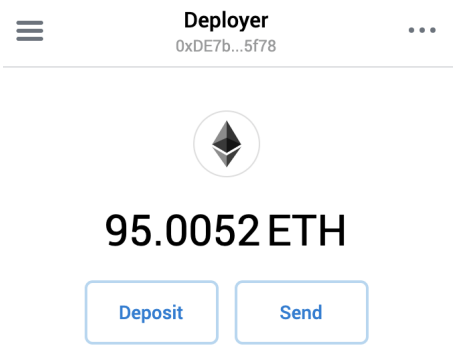

Figure 19: Deployer MetaMask

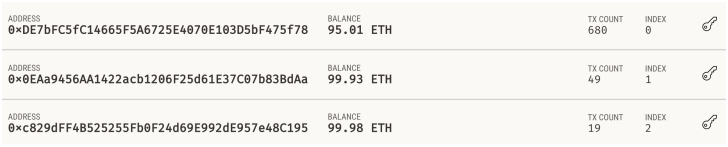

Figure 20: Deployer & Voter Ganache

3.2 Application Interface

On the welcome page, users can find instructions for using this application. Users who are not familiar with straight or cumulative voting can browse Wikipedia through links under 'About' section in Figure 21.

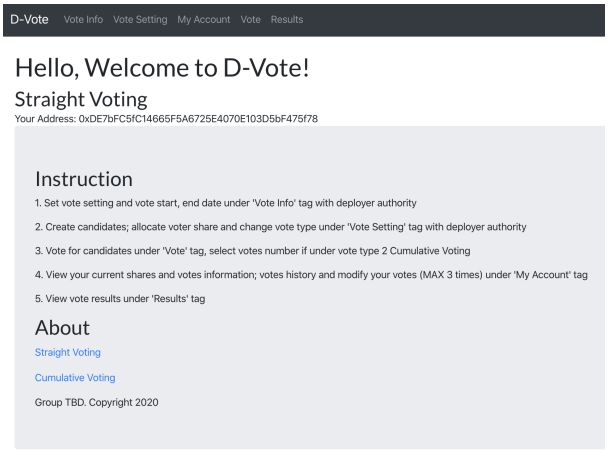

Figure 21: Welcome page

On the vote information page shown in Figure 22, users can see the general information of this vote, vote setting period, vote period. Users can also check if they can change

vote setting and if they can vote today, reset the date to default, and change current vote type. However, normal voters can only see the valid date period and check if they can vote today.

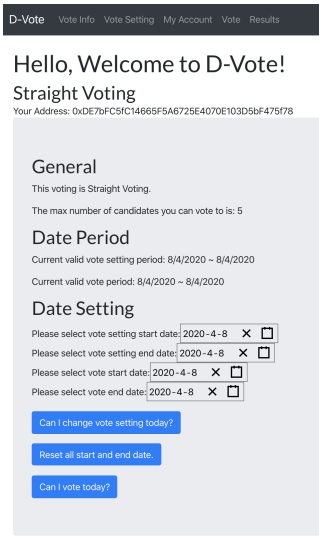

Figure 22: Vote information page

On the vote setting page shown in Figure 23, vote deployer can allocate shares among all voters and create candidates with name, photo URL, and general information. Normal voters cannot do anything on this page.

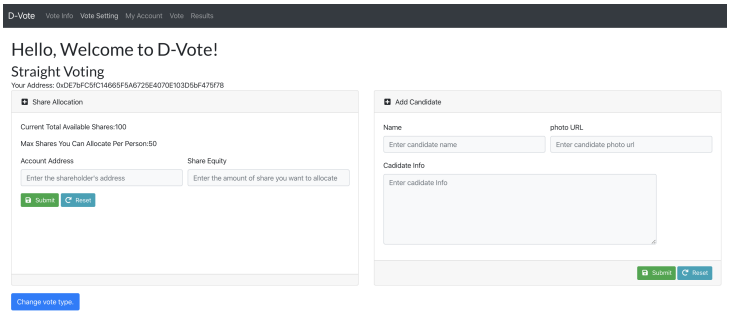

Figure 23: Vote setting page

On the my account page shown in Figure 24, users can check current account status (total shares, available votes, vote used) as well as voting records, and change votes.

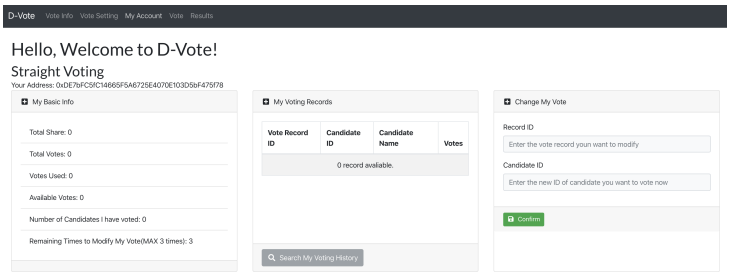

Figure 24: My account page

On the vote page shown in Figure 25, users can vote for candidates.

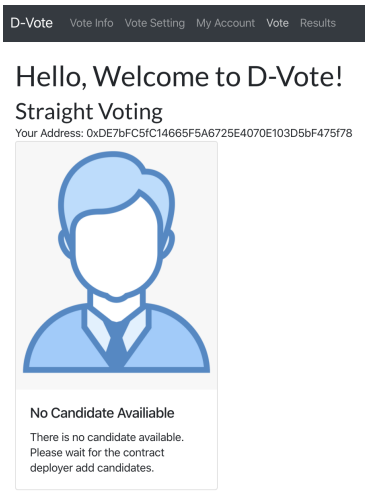

Figure 25: Vote page

On the result page shown in Figure 26, users can view real-time voting results.

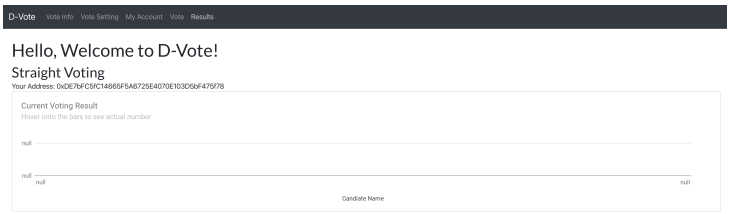

Figure 26: Vote results page

3.3 User Operations

3.3.1 Date Setting

Users can set date period using calendar tool as Figure 27 shows.

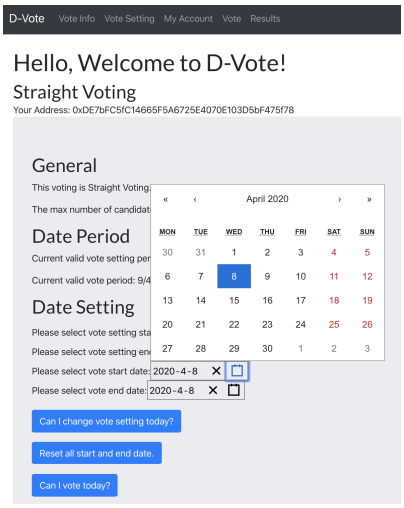

Figure 27: Date setting

Users can check whether they can change vote setting and whether they can vote on that day using the corresponding buttons shown in Figure 28 and Figure 29, and a message will appear on the top of the page for indication. Besides, users can also reset the date to default by clicking the reset button.

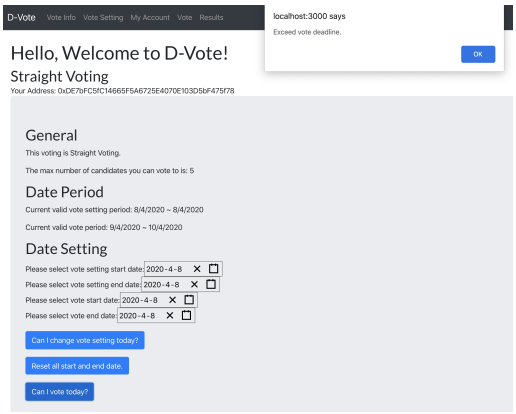

Figure 28: Vote check

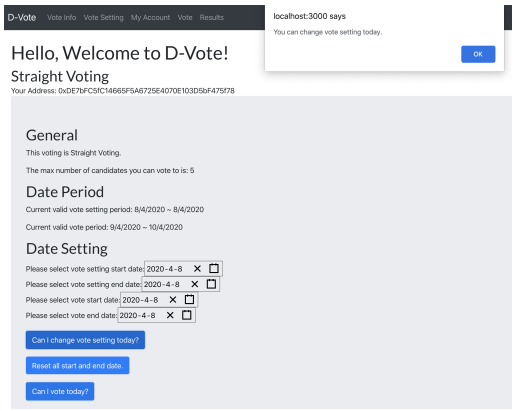

Figure 29: Vote setting check

3.3.2 Share Allocation

Vote deployer can allocate arbitrary number of shares (up to personal maximum shares) among voters by filling in voter's account address as Figure 30 shows.

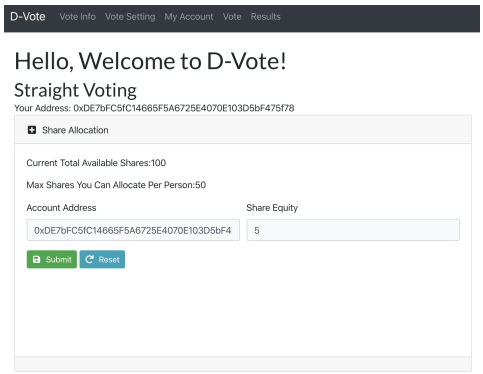

Figure 30: Share allocation

Users can check their shares, votes, and other related status under 'My Basic Info' section on the my account page shown in Figure 31.

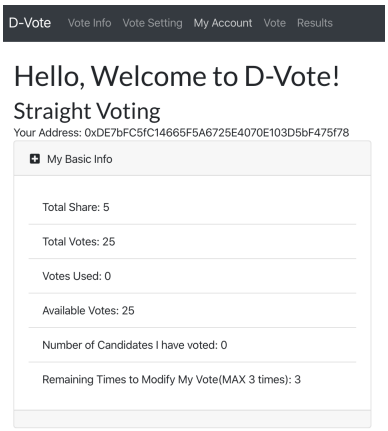

Figure 31: My basic Information after share allocation

3.3.3 Candidate Creation

Vote deployer can create candidates with name, candidate information, and photo URL shown in Figure 32 and Figure 33.

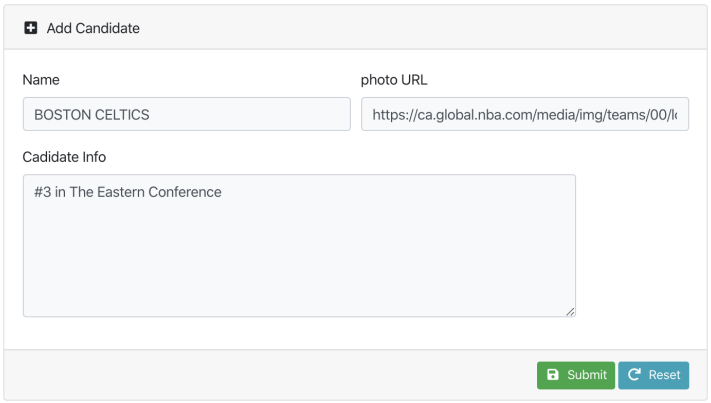

Figure 32: Candidate creation and corresponding information

Figure 33: Vote page after candidate creation

3.3.4 Vote for Candidate

Users can vote for candidates by clicking the vote button as Figure 34 shows. If it is a cumulative vote shown in Figure 35 and Figure 36, users need to input the number of votes as well. After that, users can check the voting result as Figure 37 shows.

Figure 34: After vote for candidate 1

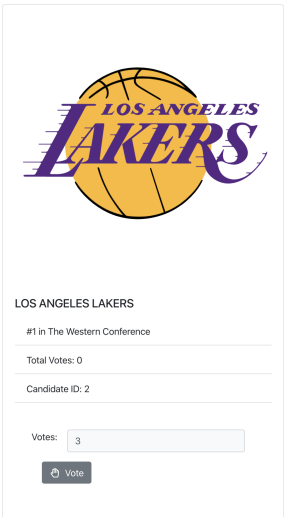

Figure 35: Vote for candidate 2 under cumulative voting

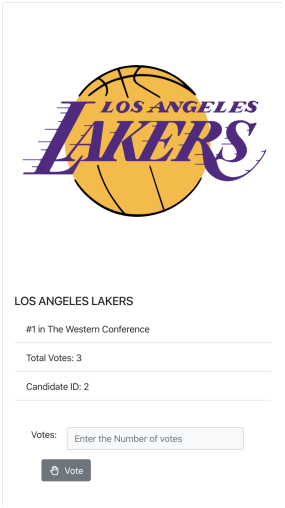

Figure 36: After vote for candidate 2

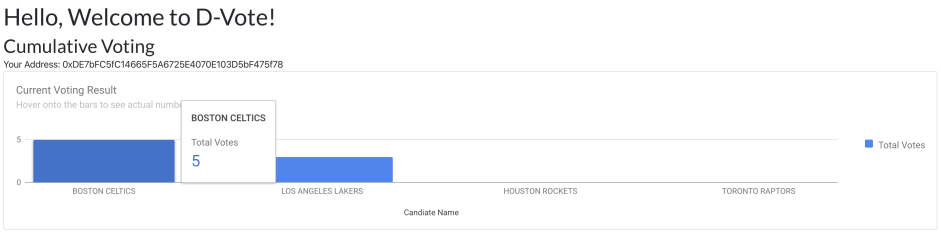

Figure 37: Result after 2 votes

3.3.5 Vote History

Users can search current voting records on the my account page. Sometimes it cannot get users the correct response for that it may not be updated. We encourage our users to wait for seconds instead of immediately click the search button as Figure 38 shows.

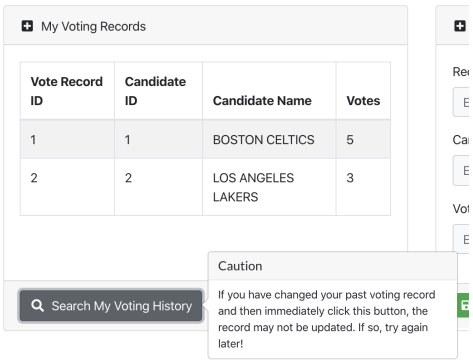

Figure 38: Vote report

3.3.6 Vote Change

Users can change vote by filling in vote record ID, new candidate ID, as well as the number of votes they would like to vote for the new candidate as Figure 39 shows.

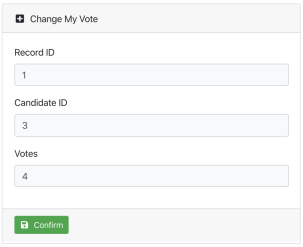

Figure 39: Vote change

3.3.7 Voter Client

A normal voter cannot do any change to the vote setting as Figure 40 shows.

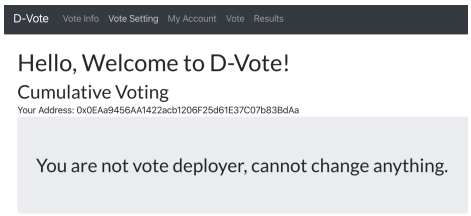

Figure 40: Vote change

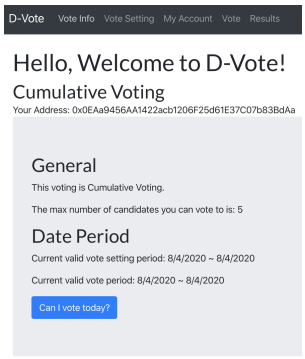

Figure 41: Vote change

4 Technical Problems & Solutions

4.1 Variable Passing Problem

Problem Description

The web app has a root component 'App' and multiple child components. The problem is how to let a child component pass a variable value back to parent component, so that the parent component can pass this value to other child components.

Solution

Considering that child component cannot pass value back to parent component in React, we construct a function that can modify the variable in the parent component. By passing this function to the child component, the child component can call the function to assign the new value to the variable. Since this variable is stored in the parent component, it avoids the process of passing back to the parent component.

4.2 Page Refresh Problem

Problem Description

The web app has a function for defining the valid vote period. This period is stored in the state of the 'App' component. Each time we switch the user account in MetaMask, the whole page will be refreshed. So after the deployer set the data and login to another account, the date will be refreshed as well. The problem is how to avoid refreshing some properties of the state when refreshing the page.

Solution

We can take advantage of local storage, which is a web API native to modern web browsers. It allows websites/apps to store data (simple and limited) in the browser, making data available in future browser sessions. We can call `localStorage.setItem(key, value)` to store data in local storage. When we need to access the data, we can call `localStorage.getItem(key)` to retrieve the value.

4.3 Result Visualization Problem

Problem Description

The problem is how to demonstrate the voting results through a bar chart in the web app.

Solution

We use a package called 'react-google-charts'. The package has multiple types of charts and all we need to do is to pass the formatted data that fits the API. If we want to demonstrate the amount of votes for each candidate, we can build a function in the 'App' component, which can call a function that retrieves the 'Candidate' struct. Then by passing this function into the 'Result' component, the component can get all the candidate information. After that, by filtering candidate name, candidate votes, and wrangling data into desired format, we can get a bar chart.

5 Future Works

Currently, the smart contract can only support one poll and one type of share. When a corporate has multiple matters that needs to be determined at the same time, the administrator has to deploy multiple contracts and allocate shares among shareholders multiple times. In the future, our DApp will support multiple votes and reduce the redundant work for the deployer.

Also, the current contract only supports one type of shares, so the contract can only be used within one organization. In reality, it is possible that an organization needs to create multiple polls for multiple corporations. For example, Shanghai stock exchange held more than 1600 online voting for 785 exchange listed companies in 2014. Thus in the future, out DApp will be able to allocate shares held by multiple companies among shareholders and create votes for multiple companies at the same time.

However, if we implement these new features in the smart contract, the data structure and the smart contract code will be much more complex. Right now lag can be observed that client has to wait for Ganache updating the data after modifying the voting record. These new features will further increase time complexity as well as transaction fee and decrease user experience. To avoid this, we need to keep a balance between processing time and functionalities, so we decide to implement these features on front-end. By allowing poll creator to set voting properties and deploy contracts on front-end, we can keep the smart contract simple: each smart contract still only supports one voting for one corporate. Meanwhile, it is fast and easy for users of any technical levels to deploy the smart contract and initiate a poll. Besides, all the information of shareholders or candidates can be imported automatically through file, so it is convenient for an organization to initiate multiple polls at the same time.

In the future, the result page will demonstrate more details such as sex ratio and distribution among voters. Features like Elliptic-Curve Cryptography (ECC) that increases security level will also be added in the future.

6 Strategic Planning & Collaborating

6.1 Market Survey

In order to successfully push our project into the market, we first propose an initial survey on the DApp market. There are many successful pilots and implementations in the field of government election. Nasdaq tested blockchain voting in 2017 with a web-based blockchain voting program. The vote took place in Estonia, a country with nationwide internet voting known as i-Voting.[1] Blockchain-based mobile voting has also been used for small-scale government elections. The Swiss city of Zug tested mobile blockchain-based voting in a consultative vote mid-2018 for voters with digital IDs.[2] In early 2018, the office of the Secretary of State of West Virginia announced it would be piloting Voatz's blockchain-based mobile voting application for eligible overseas voters. This will be the first use of this kind of application in a U.S. federal election.[3]

6.2 Self-Positioning

Our company name will be **UBC-D-Vote** and currently we will mainly focus on startups as well as some big enterprises. Our main competitors will be Voting Machines (EVMs) and Paper Ballots, whose products have already been put on the market. However, since blockchain-based election system is still in pilot in few places as mentioned above, we still have the chance to occupy a place on this market in the long term.

According to the 2019 DApp market review, DApp projects continued to suffer from their common issues like poor user retention, high turnover rate, and difficulties of navigating user interfaces.

Our project already has a user-friendly interface. Besides, the existing voting DApps in the market do not support multiple voting types and the cumulative voting model, so the features of our DApp are unique and competitive. Except for voters of elections, our potential users are the large amount of major shareholders who create an impartial vote. When we have a relatively large user size that keeps increasing, we can develop more features to support the corresponding growing needs for companies such as working process monitoring, contract monitoring etc.

As for user retention rate, according to the 2019 DApp market review, one major reason for low user retention rate is related to the transaction fee: when there is a network congestion, the gas fee can become impractical for DApp users resulting in a significant

outflow. Thus we decide to push our project onto DAppChains to both improve efficiency and lower cost. DAppChain is an Application Specific Sidechain that runs in parallel to a mainchain. It can run a consensus algorithm that optimizes for those particular types of transactions so it is more scalable.

6.3 Step-by-Step Planning

Combining market survey and self-positioning, we have developed our strategic path as shown in Figure 42, along with the path of follower, attacker, convener, and leader development.

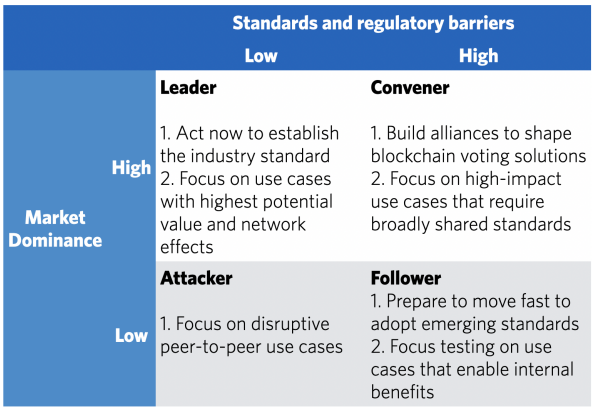

Figure 42: Strategic path

As a new Vancouver-based startup, we will keep analyzing the local market and investigating the common pain points among the local companies, by summarizing use cases that meets a minimum level of feasibility and potential return. We will keep updating the demo version and modifying features as well as functionalities based on feedback in order to earn relatively good reputation on Vancouver local market. In other words, we are going to focus on specific, promising user cases.

After the DApp has more active users, we will start looking for angel investors through online platforms like LinkedIn, Quora, and raise funds for new feature development and charge fees for these advanced functionalities. Once the DApp starts to make profit, we can attract more funds to invest the DApp for improved user experience and promoting.

Once we have built up the solid relationship with our local partners and figured out the financial problem, we will turn to our ultimate business targets – government affairs, which has the most stable and largest number of users. Once the government adopted

our DApp as its newly voting method, it means that all the voters will have to follow up and start using our products. Meanwhile, we will keep promoting this new kind of voting method to the public, making them realize it is a more transparent and cryptographic voting environment. We will have our professional promotion team, starting from district to provincial, then federal, and the whole North America market finally.

References

[1] Kalev Leetaru. How estonia's e-voting system could be the future, 2017.

[2] Sunoo Park, Michael Specter, Neha Narula, and Ronald L Rivest. Going from bad to worse: from internet voting to blockchain voting, 2020.

[3] Michael A Specter, James Koppel, and Daniel Weitzner. The ballot is busted before the blockchain: A security analysis of voatz, the first internet voting application used in us federal elections. *Preprint available at: https://internetpolicy. mit. edu/wp-content/uploads/2020/02/SecurityAnalysisOfVoatz_Public. pdf*, 2020.