Walchand College Of Engineering, Sangli.

(An Autonomous Institute)

Department
Of
Computer Science and Engineering

TY CSE Mini Project- I
Report On

Preferential Voting Using Blockchain

Submitted by

Mr. Abhishek Parmanand Charpale	**2019BTECS00071**
Mr. Kunal Santosh Kadam	**2019BTECS00064**
Mr. Prathmesh Basaweshwar Killedar	**2019BTECS00070**

Under the Guidance
of

Prof. Nandeeni Mudegol
Guide
Computer Science & Engg. Dept,
WCE, Snagli

2021-2022

Walchand College of Engineering, Sangli

(An Autonomous Institute)

Department
Of
Computer Science and Engineering

CERTIFICATE

This is to certify that the Project Report entitled, **"Preferential Voting Using Blockchain"** submitted by Mr. Charpale Abhishek, Mr. Prathmesh Killedar, Mr. Kadam Kunal, to Walchand College of Engineering, Sangli, India, is a record of bonafide Project work of course *"Mini Project 1"* carried out by them under our supervision and guidance and is worthy of consideration for the a ward of the degree of Bachelor of Technology in Computer Science & Engineering of the Institute.

Prof. Nandeeni Mudegol
Guide
Computer Sci. & Engg. Dept,
WCE, Sangli.

Dr. M. A. Shah
Head Of Department
Computer Sci. & Engg.Dept,
WCE, Sangli

Acknowledgement

We have taken effort in this project. However, this would not have been possible without the kind support and help of many individuals and organisations. I would Like to Extend my sincere thanks to all of them.

We are highly indebted to Prof. P. H. Sawant(Director), Dr. M. A. Shah(HOD), Mrs. Nandeeni Mudegol(Project Coordinator) for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

We would also like to thank the teammates who worked together in finishing this project within the limited time. Finally, thanks to all who supported the project.

Declaration

I hereby declare that work presented in this project report titled **"PREFERENTIAL VOTING USING BLOCKCHAIN"** submitted by us in the partial fulfillment of the requirement of the award of the degree of **Bachelor of Technology (B.Tech)** Submitted in the **Department of Computer Science & Engineering, Walchand College of Engineering, Sangli**, is an au-thentic record of our project work carried out under the guidance of Prof. Nandeeni Mudegol.

Date: 24-11-2021

Place : Sangli

(Signature)
Mr. Abhishek Charpale
(2019BTECS00071)
Mr. Prathmesh Killedar
(2019BTECS00070)
Mr. Kunal Kadam
(2019BTECS00064)

Table Of Contents

1 Project title

Preferential Voting Using Blockchain

2 Abstract

In the current system, voting is done by using EVM(Electronic Voting Machine)

This system can be replaced by the online voting (E-voting) system which will limit voting frauds.

Expanding e-voting into Blockchain technology could be the solution to alleviate the present concerns in E-voting.

In a survey we found that sometimes elected candidates have a minor percentage of votes of total votes which is sometimes not appreciated by everyone.

So to have a majority of votes and have a fair election Preferential Voting is a good solution.

With this view in mind we are going to develop an Online Preferential Voting System using Blockchain.

In this voting the admin has the access to add the candidates who stand in the election, if the admin has the metamask and has the right to modify. Also the admin has to give voting rights to the voters by adding the metaskmask id of the voters.

The screen provided to the voters shows them the candidates standing for the election and provides a checkbox for giving the preferences of the voters.

The winner screen provided gives us the winners name after the election is over.

This E-voting system has the potential to make the voting process easier and more accessible for electors.

This makes elections much more fair than the traditional method.

3 Introduction and Related work

In a survey we found that sometimes elected candidates have a minor percentage of total votes which is sometimes not appreciated by everyone. So to have a majority of votes and have a fair election, Preferential Voting is a good solution. In this voting the voters can give their preferences of their choice and according to the preferences we can select the winner of the election. This makes elections much more fair than the traditional method.

4 Problem statement

Create a Preferential Voting App using Ethereum Blockchain

5 Objectives

Requirement Gathering:

- To design a web application for preferential voting using blockchain and metamask

Data analysis :
https://policyreview.info/articles/analysis/voter-preferences-voter-manipulation-voter-analytics-policy-options-less

https://towardsdatascience.com/visualisation-of-ranked-choice-voting-in-r-888f51134870

- The web application have the following features

1. Voters can generate a preference list of candidates which is cast as a vote
2. In this case, if a candidate has more than 50% of first preference votes, the candidate wins.
3. The candidate with lowest first preference votes is eliminated and his votes are redistributed to other candidates.
4. First, sort votes according to voter's first preference
5. Count the number of first preferences votes for each candidate and record these totals in the 1st count of the tally sheet.
6. To be elected the candidate should have more than half of the votes.
7. Candidate with the fewest votes is excluded and the votes are redistributed to other candidates as per their second preferences.
8. After this we get the 2nd count total. We check if there is a majority of votes to any candidate then he is declared as the winner.
9. If not then the steps of excluding the candidate is repeated so that we get the winner having the majority of votes i.e. more than 50%.

6 Methodology

- Smart Contract:

  Basically speaking, smart contracts are programs that are executed on Ethereum platform. Ethereum Blockchain is a decentralized computing platform that not only provides a platform for crypto-currency transactions but also allows execution of certain programs called smart contract executes such programs. Execution of such decentralized applications on Ethereum platform is possible using a virtual machine called Ethereum Virtual Machine

  The state of a smart contract can be changed in Ethereum blockchain, but to perform some changes, some fees or fuel is required. This fee or fuel or gas in Ethereum is called Ether. Thus, we can say that Ether is the fuel for operating this decentralized Ethereum platform.

- BlockChain

  Blockchain can be thought of as a chain of blocks and these blocks contain transactions' information. If we break the word Blockchain, then the word "block" in this context means the digital information or transactions stored in the form of linked list, the "chain". Basically, the blocks store information about transactions like the amount, timestamp, and other information if we talk about the case of crypto-currency. Blocks also store information that helps in distinguishing one block from another.

  With other information in a block, a block also has a hash. All the contents of the block as well as the block itself can be uniquely identified by this hash. The hash works in such a way that once a block is created by adding transactions to it, any change, in fact, even a single character change in the data of a block causes the hash value of that block to change. Once a block's hash value changes, it does not remain the same block again. We can say that each block basically has 3 parts or we can say that each block stores the following information. These are Data, Hash of previous block and Hash.

- BlockChain Architecture

  Blockchain is an architecture comprising multiple components and what makes blockchain unique is the way these components function and interact with each other. Some of the important Ethereum components are Ethereum Virtual Machine (EVM), miner, block, transaction, consensus algorithm, account, smart contract, mining, Ether, and gas. We are going to discuss each of these components in this chapter.

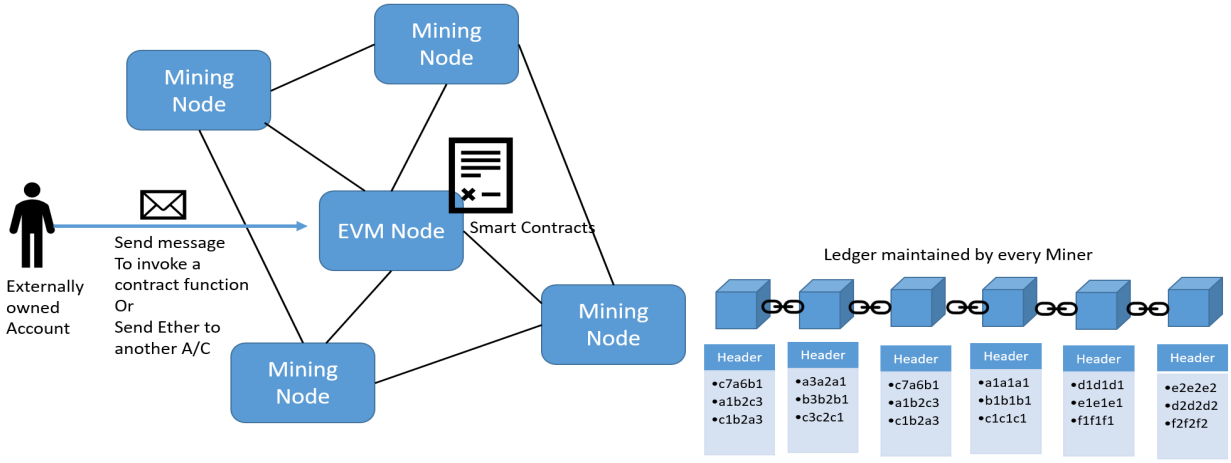
A blockchain network consists of multiple nodes belonging to miners and some nodes that do not mine but help in execution of smart contracts and transactions. These are known as EVMs. Each node is connected to another node on the network. These nodes use peer-to-peer protocol to talk to each other. They, by default, use port 30303 to talk among themselves.

Each miner maintains an instance of ledger. A ledger contains all blocks in the chain. With multiple miners it is quite possible that each miner's ledger instance might have different blocks to another. The miners synchronize their blocks on an on-going basis to ensure that every miner's ledger instance is the same as the other.

Details about ledgers, blocks, and transactions are discussed in detail in subsequent sections in this chapter.

The EVM also hosts smart contracts. Smart contracts help in extending Ethereum by writing custom business functionality into it. These smart contracts can be executed as part of a transaction and it follows the process of mining as discussed earlier.

A person having an account on a network can send a message for transfer of Ether from their account to another or can send a message to invoke a function within a contract. Ethereum does not distinguish them as far as transactions are considered. The transaction must be digitally signed with an account holder's private key. This is to ensure that the identity of the sender can be established while verifying the transaction and changing the balances of multiple accounts. Let's take a look at the components of Ethereum in the following diagram:



- So the working of our project is as follows

  So the framework used in our project is the next js.

  The blockchain used by us is the ethereum blockchain.

  To use/access the ethereum blockchain we have to create the metamask account.

  The algorithm that used to calculate the results is as follows:

```
pragma solidity ^0.4.17;
 contract PreVoting{


     //information about the candidates standing in election
     struct Candidate{
         string candidate_name;
//name of the candidates or the party
         uint vote_count;
//vote count for this candidate
         address[] address_of_voters_for_this_candidate;
     }

     struct Voter{
         bool voted;
//has voter already voted
         bool has_voting_rights;
//does the voter has voting rights?
         uint [] voting_preferencing;
//this array will store the preference list of each voter

         uint redistributing_index;
     }

     address public organiser;
//address of the organiser/owner/admin of the contract
```

```
    uint public majority_percentage;
//it is the minimum percentage of votes required to win the elections

    string public election_name;
//as the voting can be used for different elections we need to give
the election name

    mapping(address => Voter) public voters;
//key vantage pair so that each voter can be mapped with address to
Voter Structure

    Candidate[] public candidates;
//list of all the candidates who stood up for voting

    address [] public address_of_voters;

    string public winner;

    function PreVoting() public {
       organiser = msg.sender;
       election_name = "Pre_votig";
       majority_percentage = 50;
    }


   //function to add candidate for elections
    function add_candidate(string cname) public{
        require(msg.sender==organiser);
       //  ,"Only organiser is allowed to add candidates"
        candidates.push(Candidate(cname,0,new address[](0)));
    }


    //function to set majority if required to change majority
    function set_majority(uint majority)public{
        require(msg.sender==organiser);
       //  ,"Only organiser is allowed to set the winning majority"
        majority_percentage=majority;
    }
```

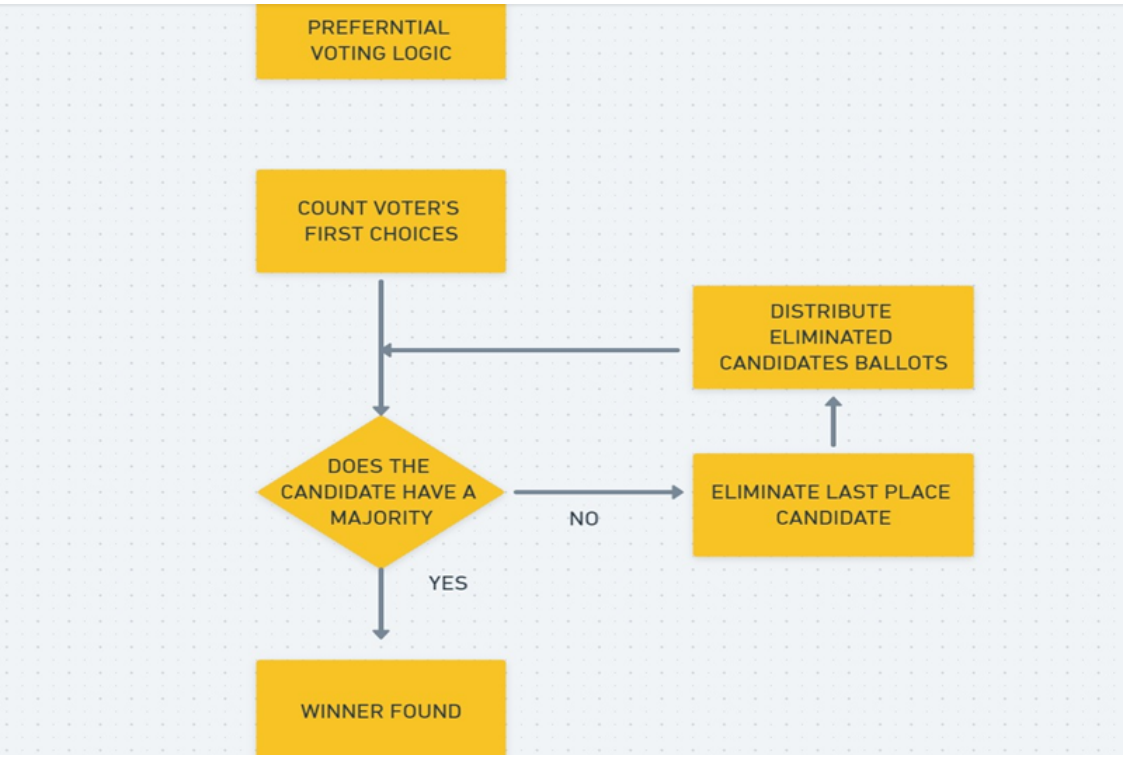
```
    //functionto get number of candidates stood for elections
    function get_number_of_candidates()public view returns(uint){
        return candidates.length;
    }


    function get_number_of_voters_who_voted() public view returns
(uint){
        return address_of_voters.length;
    }

    function getCandidatesNames(uint pos)public view returns(string){
        return candidates[pos].candidate_name;
    }


    uint public max_per;


    //function to check majority for each cycle of the calculation
phase
    function check_majority() public{
        uint
number_of_voters_who_voted=get_number_of_voters_who_voted();
        uint max_vote_count = 0;
        uint min_vote_count = number_of_voters_who_voted;
        uint max_voted_candidate;
        uint min_voted_candidate;

        uint number_of_candidates=get_number_of_candidates();

        for(uint i=0;i<number_of_candidates;i++){
            if(candidates[i].vote_count > max_vote_count){
                max_vote_count = candidates[i].vote_count;
                max_voted_candidate=i;
            }
```

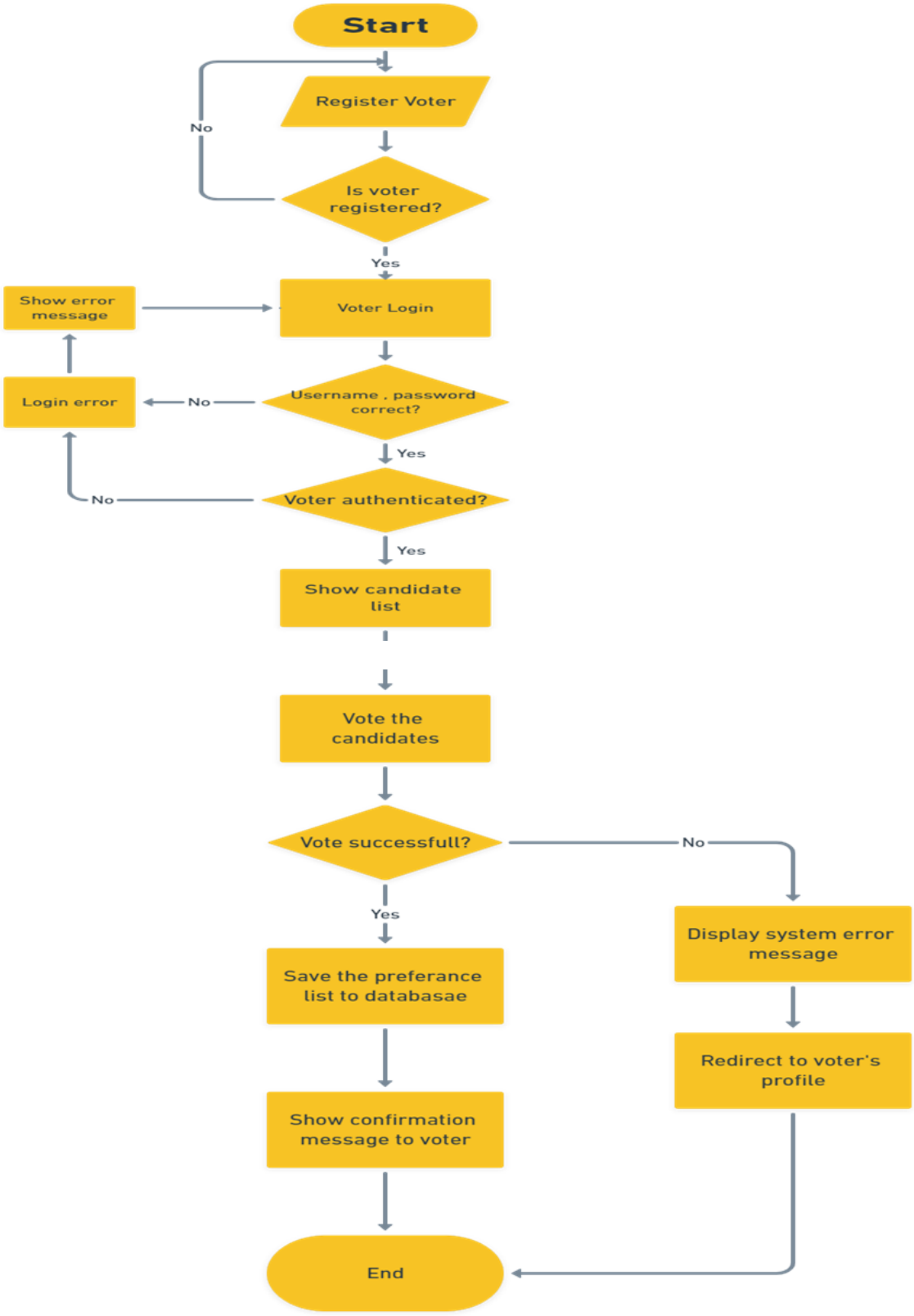
```
            if(candidates[i].vote_count<min_vote_count){
                min_vote_count=candidates[i].vote_count;
                min_voted_candidate=i;
            }
        }

        uint max_candidate_vote_percentage =
max_vote_count*100/number_of_voters_who_voted;

        if(max_candidate_vote_percentage>=majority_percentage){
            winner=candidates[max_voted_candidate].candidate_name;
        }else{
            redistribute_votes(min_voted_candidate);
        }

    }


    // function to give voting rights to voters
    function give_voting_rights(address voter)public{
        require(msg.sender==organiser);
       //  ,"Only organiser is allowed to give voting rights"
        voters[voter].has_voting_rights=true;
    }



    function vote(uint[] preference_array) public {
        require(voters[msg.sender].has_voting_rights==true);
       //  ,"You are not authorised to vote"
        require(voters[msg.sender].voted==false);
       //  "You have already casted your vote"
        require(preference_array.length==get_number_of_candidates());
       //  ,"Please enter the exact numbers of that candidates"

        for(uint i=0;i<preference_array.length;i++){
           preference_array[i]--;
        }
```

```
        voters[msg.sender].voting_preferencing=preference_array;
        voters[msg.sender].voted=true;

        uint candidate_index =
voters[msg.sender].voting_preferencing[0];              //get the
candidate who has received the highest priority

candidates[candidate_index].address_of_voters_for_this_candidate.push(
msg.sender);

        address_of_voters.push(msg.sender);
    }

    // function to calculate votes to each candidate(first priority)
    function calculate_votes() public {
        uint number_of_voters_who_voted=
get_number_of_voters_who_voted();

        for(uint i=0;i<number_of_voters_who_voted;i++){
            address address_of_voter=address_of_voters[i];
            uint
candidate_index=voters[address_of_voter].voting_preferencing[0];

            candidates[candidate_index].vote_count--;
        }
    }

    function redistribute_votes(uint min_voted_candidate)public{
        uint voters_for_min_vited_candidates =
candidates[min_voted_candidate].address_of_voters_for_this_candidate.l
ength;
        for(uint i=0;i<voters_for_min_vited_candidates;i++){
            address voter_address =
candidates[min_voted_candidate].address_of_voters_for_this_candidate[i
];
            voters[voter_address].redistributing_index++;
            uint candidate_index_to_vote =
voters[voter_address].voting_preferencing[voters[voter_address].redist
ributing_index];
```

```
            candidates[candidate_index_to_vote].vote_count++;
        }

candidates[min_voted_candidate]=candidates[candidates.length-1];
        candidates.length--;
    }
 }
```

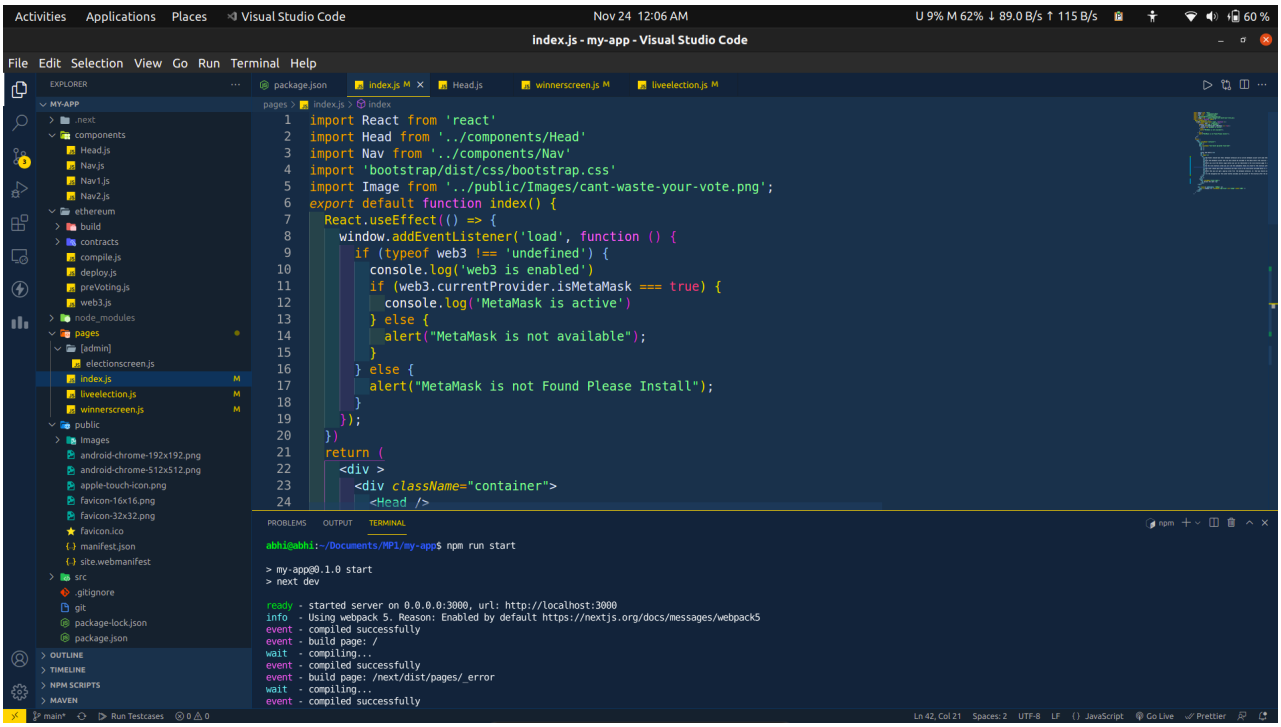
- **Project diagrams**




Start
Register Voter
No
Is voter registered?
Yes
Show error message
Voter Login
Login error
No
Username , password correct?
Yes
No
Voter authenticated?
Yes
Show candidate list
Vote the candidates
Vote successfull?
No
Yes
Display system error message
Save the preferance list to databasae
Redirect to voter's profile
Show confirmation message to voter
End

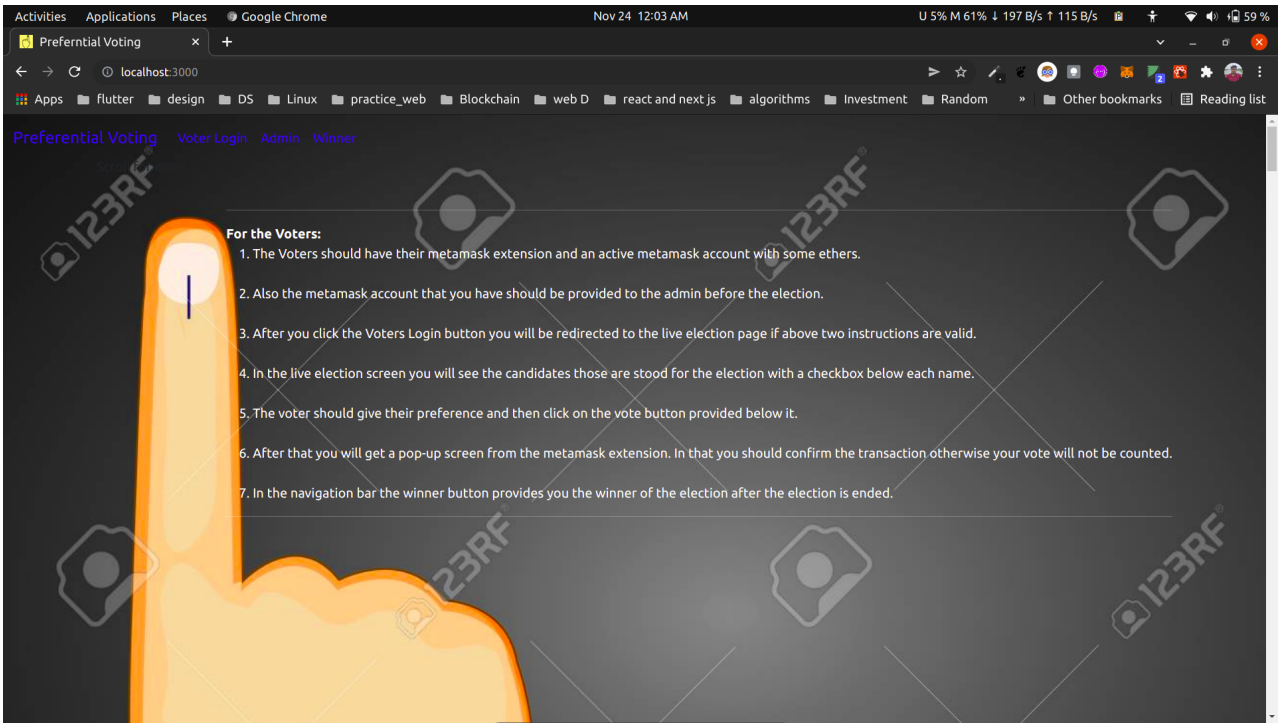
• Testing (Unit, Integration and System)

- **Unit Testing**: The unit testing of components, functions, classes have been done manually.

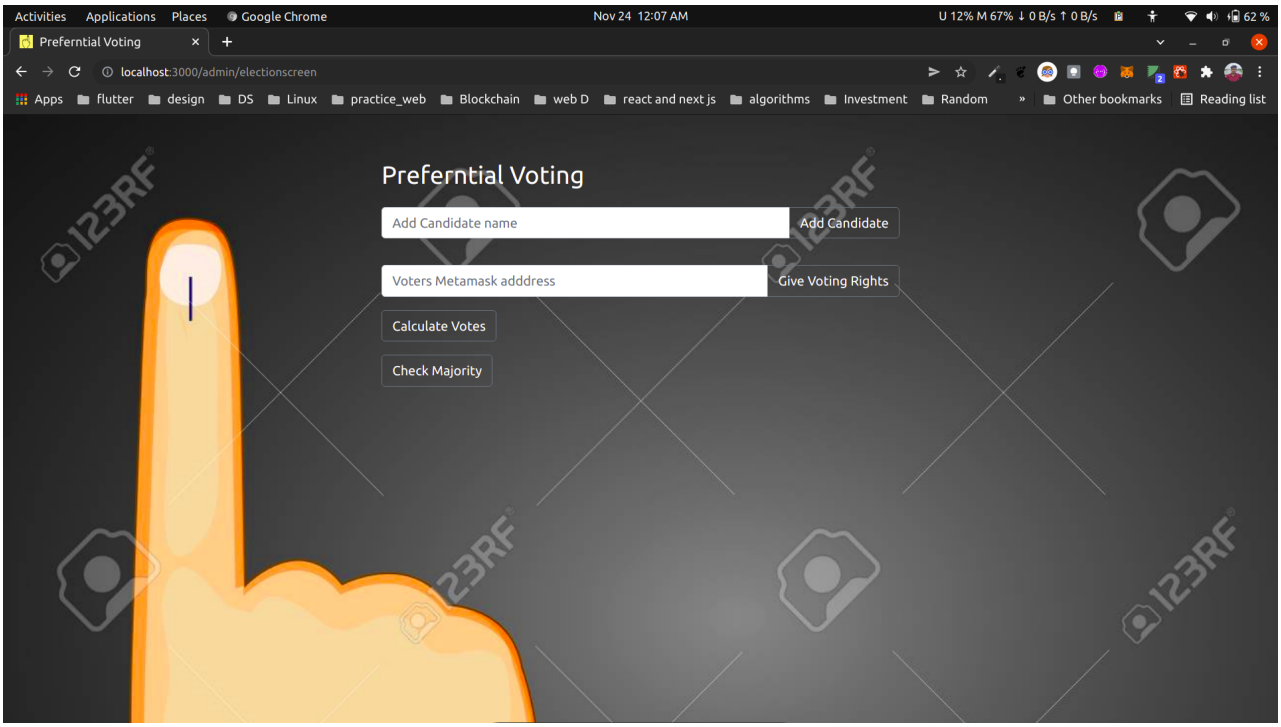


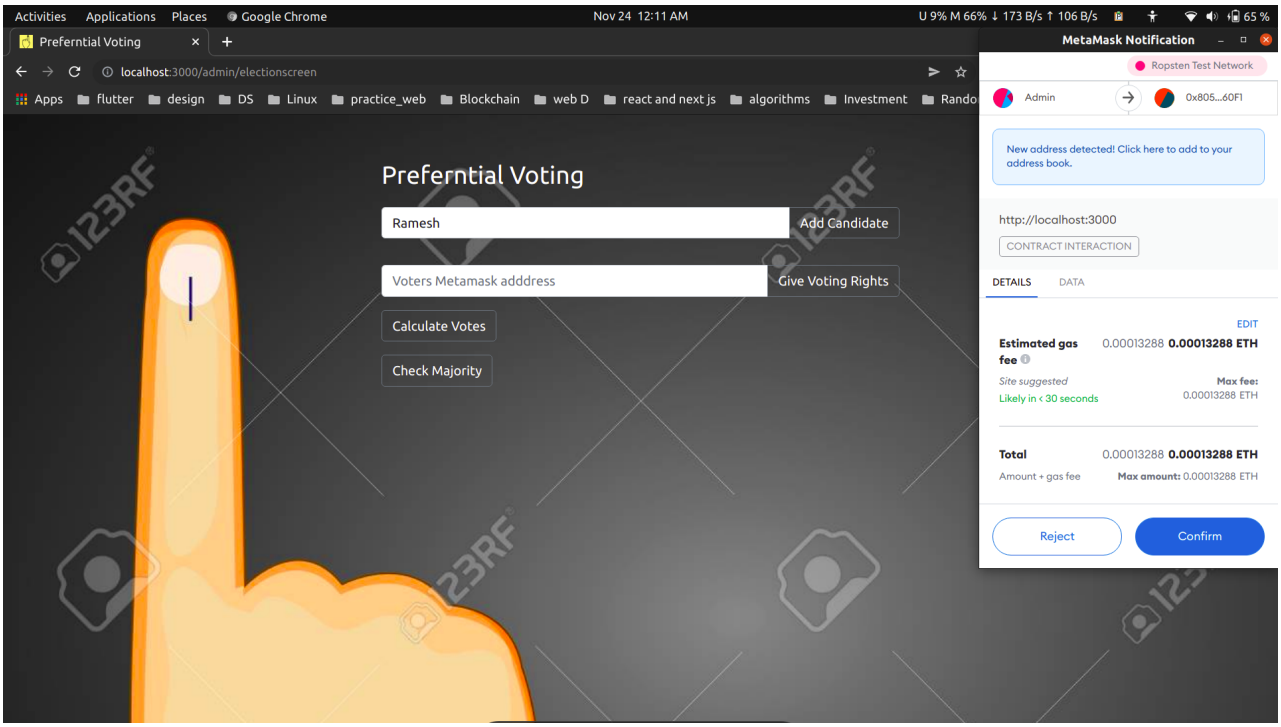
- **Integration testing**
  After completion of the project , manually , integration testing was done . Found no-errors after combining all the modules. Following are the snapshots :

Dashboard / Homepage:

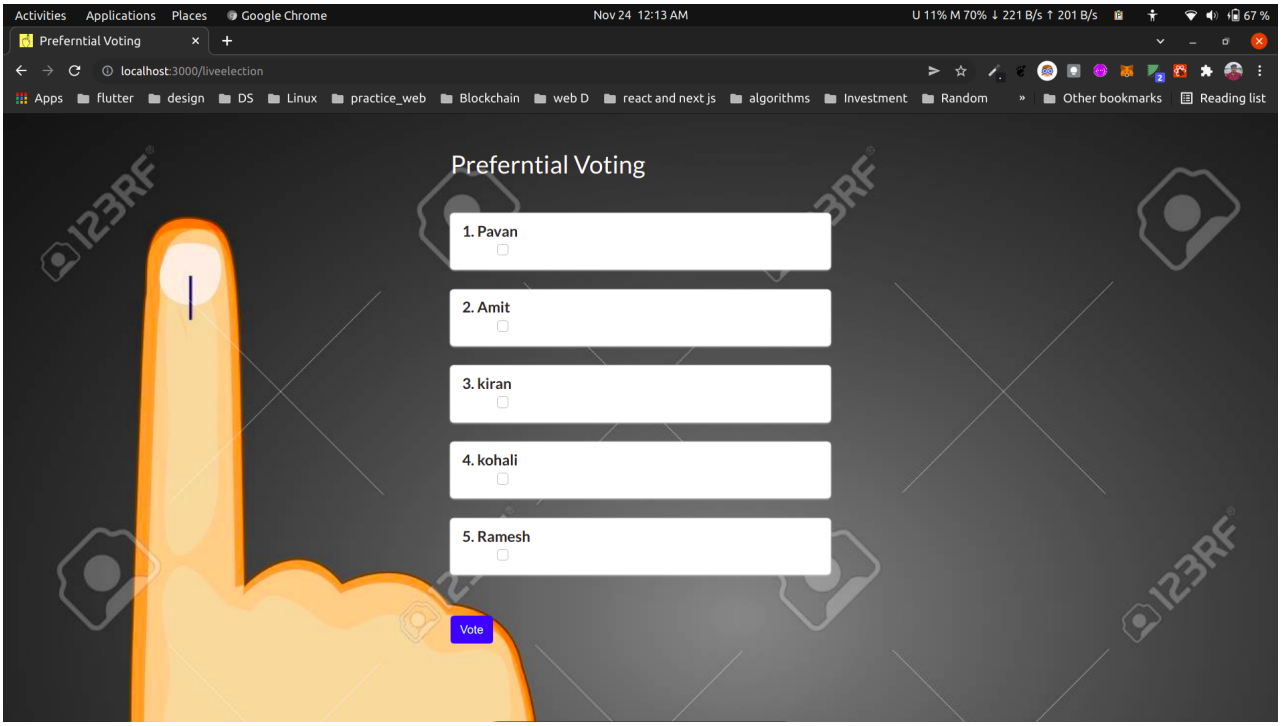


Admin Side:



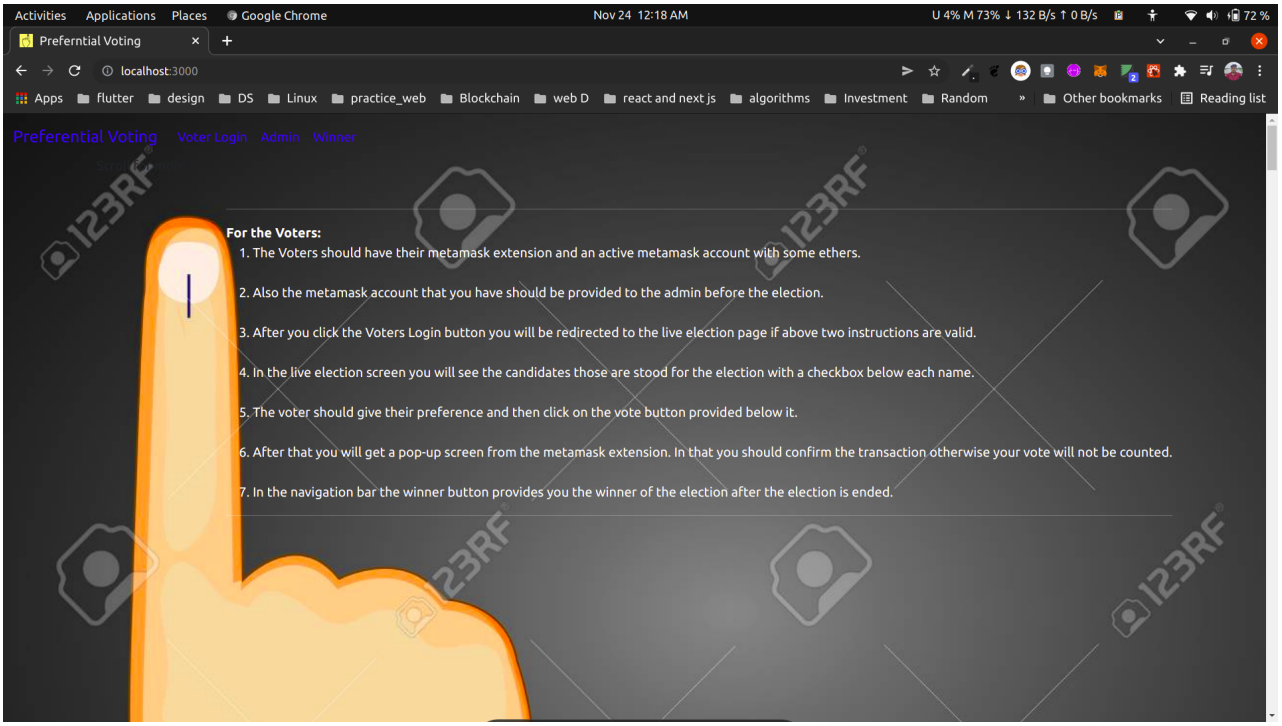

Activities
Applications
Places
Google Chrome
Nov 24 12:11 AM
Preferntial Voting
localhost:3000/admin/electionscreen
Preferntial Voting
Ramesh
Add Candidate
Voters Metamask adddress
Give Voting Rights
Calculate Votes
Check Majority
MetaMask Notification
Ropsten Test Network
Admin
0x805...60F1
New address detected! Click here to add to your address book.
http://localhost:3000
CONTRACT INTERACTION
DETAILS
DATA
EDIT
Estimated gas fee
0.00013288 0.00013288 ETH
Site suggested
Likely in < 30 seconds
Max fee:
0.00013288 ETH
Total
0.00013288 0.00013288 ETH
Amount + gas fee
Max amount: 0.00013288 ETH
Reject
Confirm

Voters side -

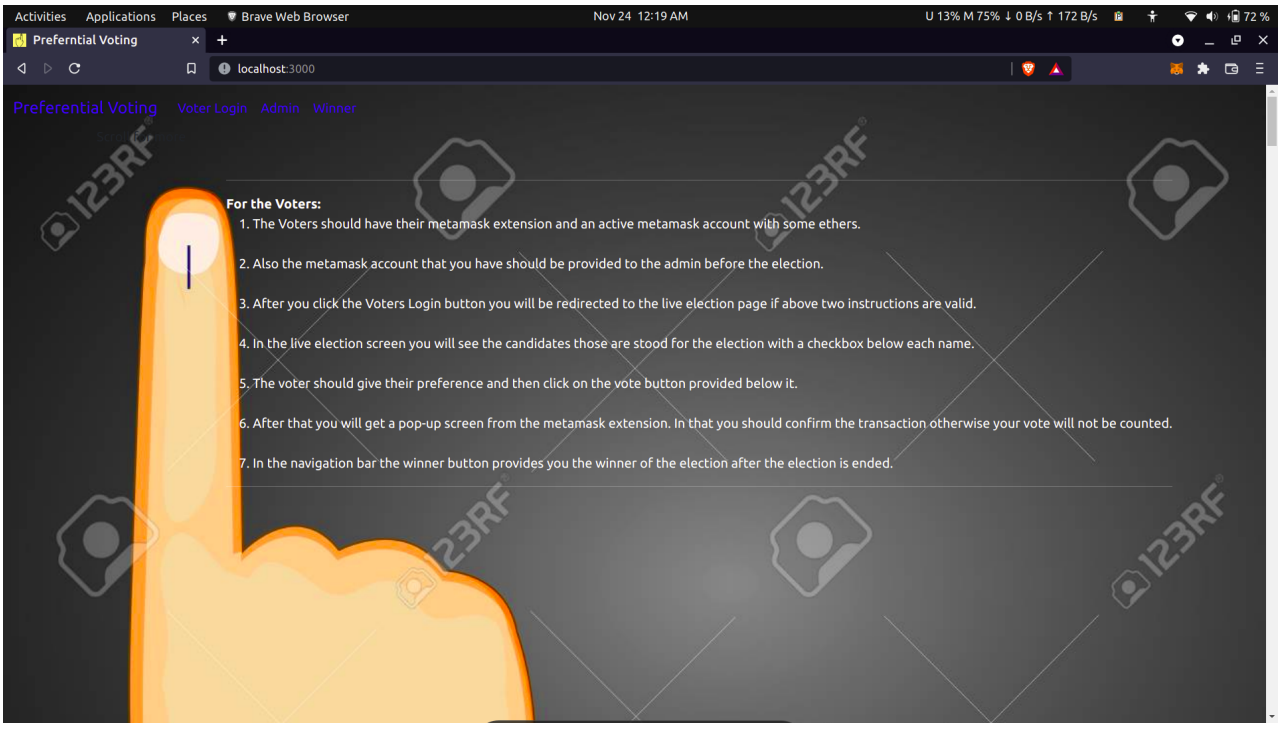

Activities
Applications
Places
Google Chrome
Nov 24 12:13 AM
Preferntial Voting
localhost:3000/liveelection
Preferntial Voting
1. Pavan
2. Amit
3. kiran
4. kohali
5. Ramesh
Vote

winner Screen:

- **System Testing**
  The application works perfectly on all browsers. This testing is done on Chrome and Brave browser.
  Here are the snapshots :
  Chrome:


Activities Applications Places Google Chrome Nov 24 12:18 AM
Preferntial Voting
localhost:3000
Preferential Voting Voter Login Admin Winner
For the Voters:
1. The Voters should have their metamask extension and an active metamask account with some ethers.
2. Also the metamask account that you have should be provided to the admin before the election.
3. After you click the Voters Login button you will be redirected to the live election page if above two instructions are valid.
4. In the live election screen you will see the candidates those are stood for the election with a checkbox below each name.
5. The voter should give their preference and then click on the vote button provided below it.
6. After that you will get a pop-up screen from the metamask extension. In that you should confirm the transaction otherwise your vote will not be counted.
7. In the navigation bar the winner button provides you the winner of the election after the election is ended.

Brave:

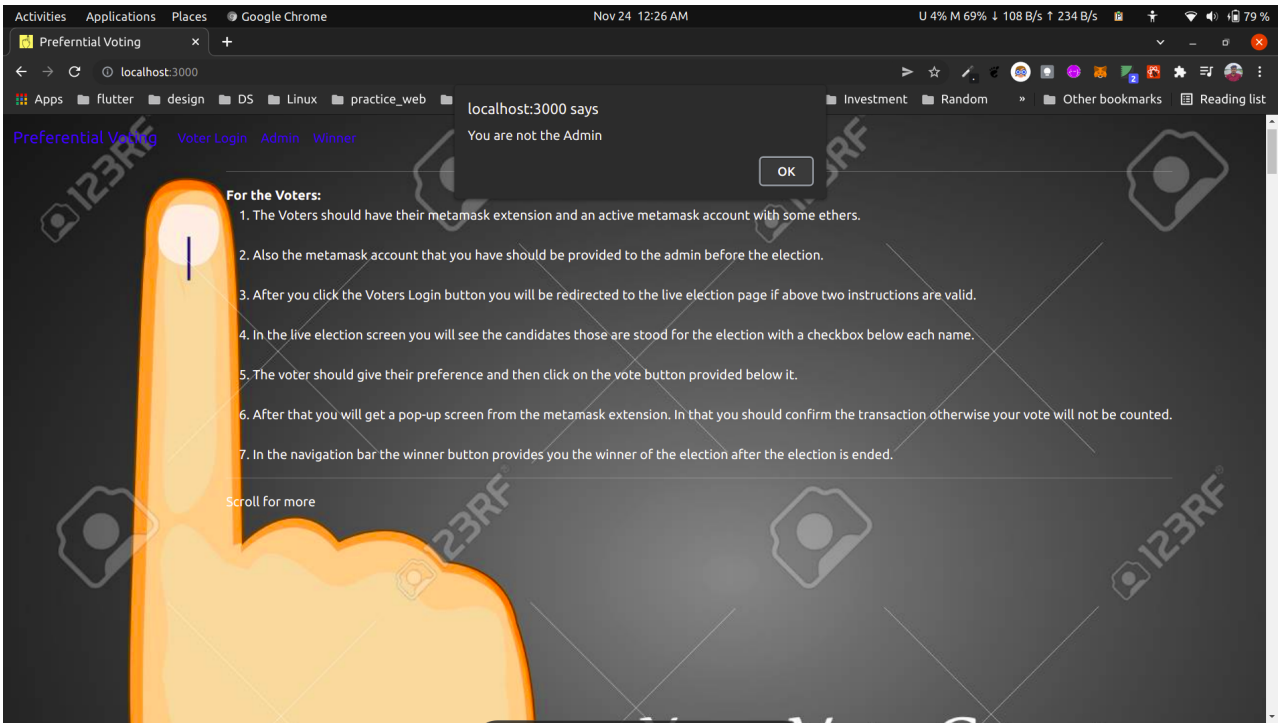

Activities Applications Places Brave Web Browser Nov 24 12:19 AM
Preferntial Voting
localhost:3000
Preferential Voting Voter Login Admin Winner
For the Voters:
1. The Voters should have their metamask extension and an active metamask account with some ethers.
2. Also the metamask account that you have should be provided to the admin before the election.
3. After you click the Voters Login button you will be redirected to the live election page if above two instructions are valid.
4. In the live election screen you will see the candidates those are stood for the election with a checkbox below each name.
5. The voter should give their preference and then click on the vote button provided below it.
6. After that you will get a pop-up screen from the metamask extension. In that you should confirm the transaction otherwise your vote will not be counted.
7. In the navigation bar the winner button provides you the winner of the election after the election is ended.

- **UI testing**
  Testing of the user interface is done manually as well as following checks were done -
  1. **Window size :**
     We tested the application on various browsers as above. All the widgets are getting properly aligned according to the screen size. For example here , we have added a few snapshots of Home Screen when the website is started on multiple browsers as shown above.

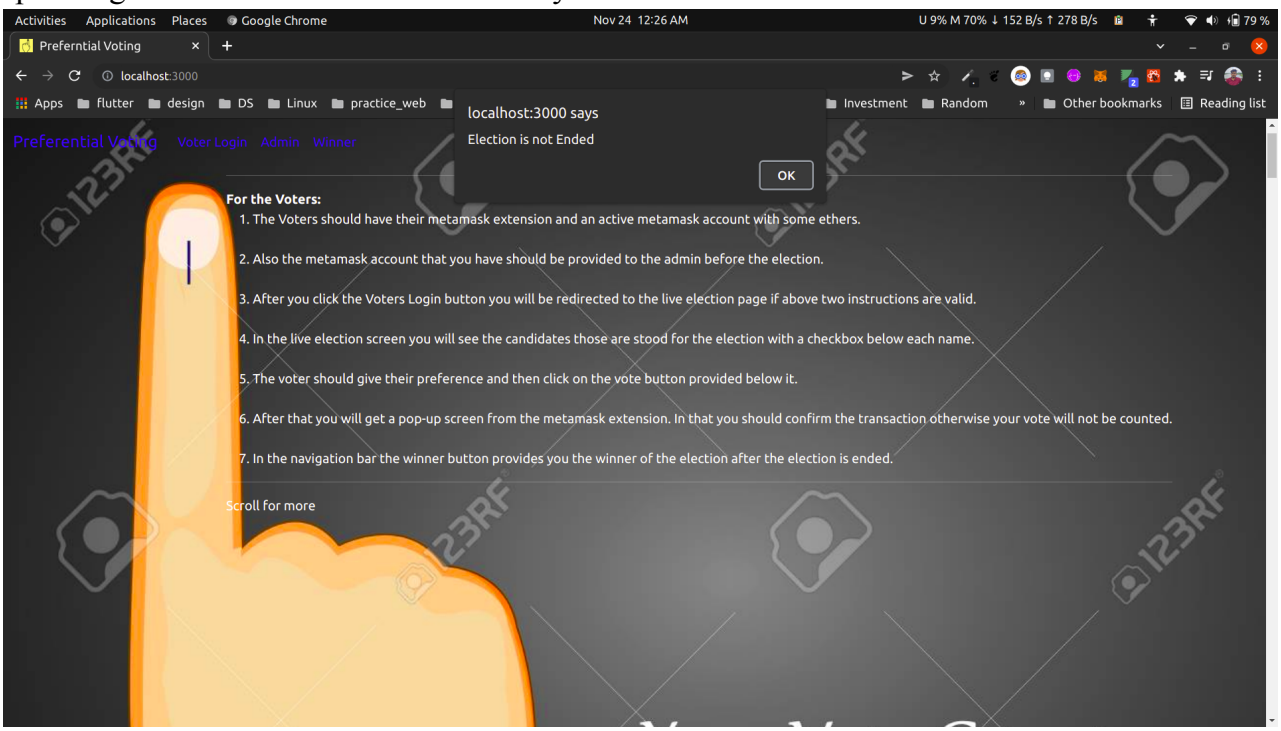
  2. **Navigation elements -** Navigators are working fine & when the respective navigational button is pressed , the user is getting redirected to the appropriate page .

  3. **Error Logging -** When the system experiences any error or delay, the application informs the user about it .

Here if the voter tried to login For Admin he/she will not have access to Admin login page so this alert pops up.



The voting process is not completed. And users want to see the winner screen so this alert pops up telling them the elections are not over yet.



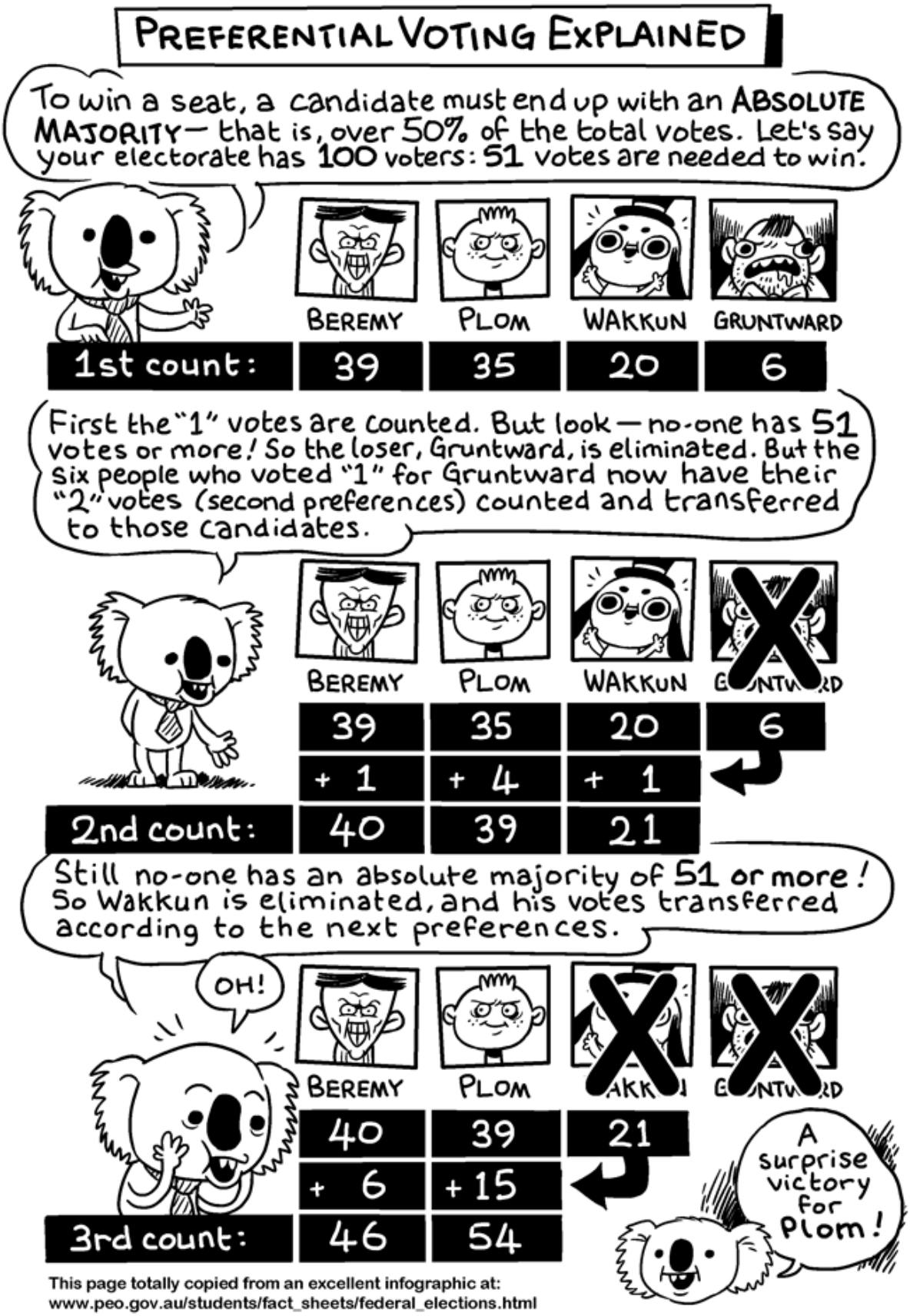
Results and Conclusion

Conclusion

- This application can be used at small scale at schools and colleges
- After the invention of the application the voting can be done in a more fair manner and user friendly way.
- There was an enhancement of the voting system making it more secure.
- Future scope of this project is

  After addition of the aadhar link to the metamask id of the voters, this application can be used for our elections held in parliament or in our country.

Example :


PREFERENTIAL VOTING EXPLAINED
To win a seat, a candidate must end up with an ABSOLUTE MAJORITY— that is, over 50% of the total votes. Let's say your electorate has 100 voters: 51 votes are needed to win.
BEREMY
PLOM
WAKKUN
GRUNTWARD
1st count: 39 35 20 6
First the "1" votes are counted. But look — no-one has 51 votes or more! So the loser, Gruntward, is eliminated. But the six people who voted "1" for Gruntward now have their "2" votes (second preferences) counted and transferred to those candidates.
BEREMY
PLOM
WAKKUN
39 35 20 6
+ 1 + 4 + 1
2nd count: 40 39 21
Still no-one has an absolute majority of 51 or more! So Wakkun is eliminated, and his votes transferred according to the next preferences.
OH!
BEREMY
PLOM
40 39 21
+ 6 + 15
3rd count: 46 54
A surprise victory for Plom!
This page totally copied from an excellent infographic at:
www.peo.gov.au/students/fact_sheets/federal_elections.html

Result

- Hosted Project Link :

  Project is hosted on vercel. which is a platform to deploy projects. Vercel is delicately made for NextJs.

  https://preferentialvoting.vercel.app/

- Github Project Link :

  https://github.com/abhi05-04/preferential_voting_Blockhain

References

- Information Regarding Preferential Voting

  https://en.wikipedia.org/wiki/Preferential_voting

  https://www.ecanz.gov.au/electoral-systems/preferential
- Website of Ethereum

  https://ethereum.org/en/
- Website of Infura to store block

  https://infura.io/
- Website of Metamask

  https://metamask.io/