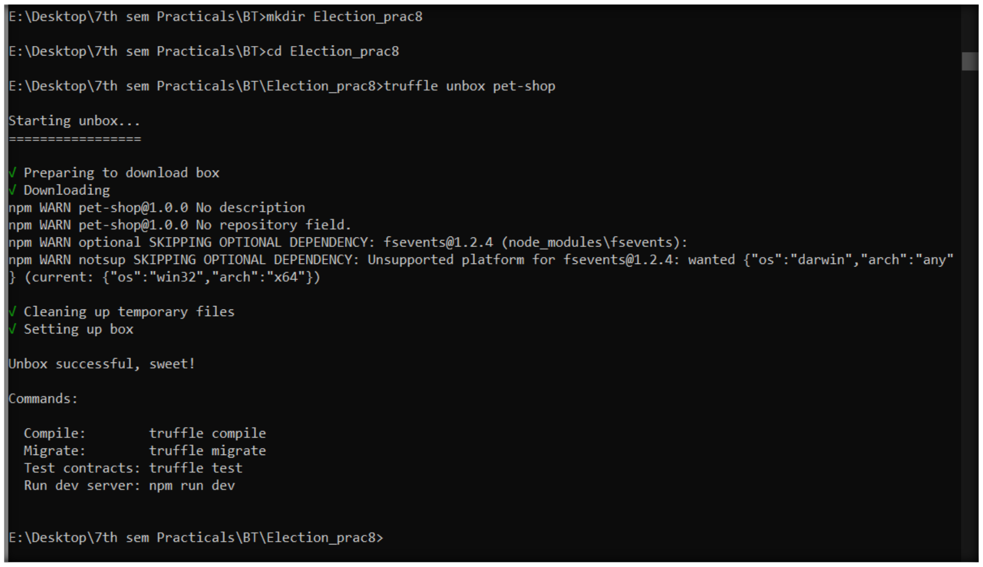
# PRACTICAL 8

## AIM:
Create Decentralized application “Voting” using Ethereum. Set up development environment using Truffle framework and Ganache, Metamask of chrome extension.

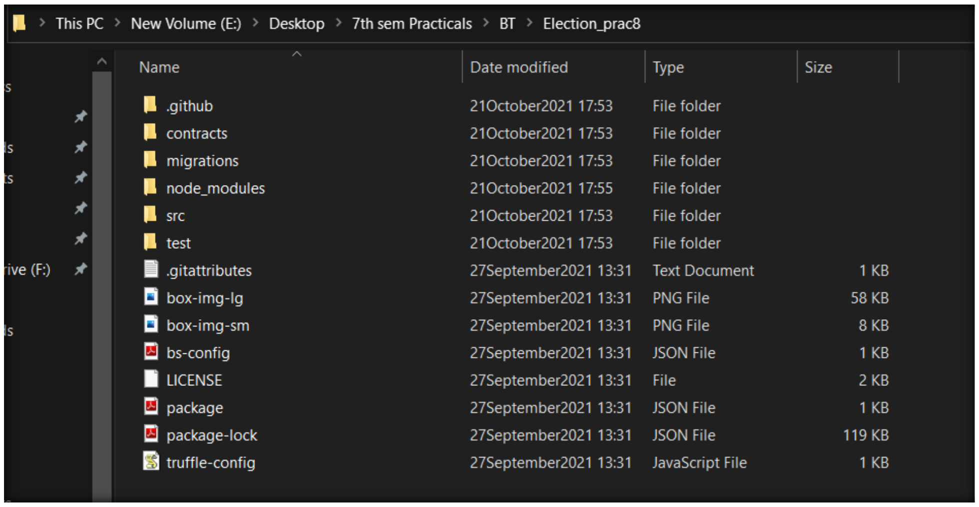
## PREREQUISITES:
- Nodejs
- Ganache
- Truffle
- Metamask Extension

## STEPS TO MAKE DECENTRALIZED ELECTION SYTEM:
1. First make an empty directory and then run below command in cmd to download and unbox pet-shop example.
    a. truffle unbox pet-shop



```
E:\Desktop\7th sem Practicals\BT>mkdir Election_prac8

E:\Desktop\7th sem Practicals\BT>cd Election_prac8

E:\Desktop\7th sem Practicals\BT\Election_prac8>truffle unbox pet-shop

Starting unbox...
=================

√ Preparing to download box
√ Downloading
npm WARN pet-shop@1.0.0 No description
npm WARN pet-shop@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch":"any"
} (current: {"os":"win32","arch":"x64"})

√ Cleaning up temporary files
√ Setting up box

Unbox successful, sweet!

Commands:

  Compile:        truffle compile
  Migrate:        truffle migrate
  Test contracts: truffle test
  Run dev server: npm run dev


E:\Desktop\7th sem Practicals\BT\Election_prac8>
```

2. now you can see the file structure like below image in that empty directory.



This PC > New Volume (E:) > Desktop > 7th sem Practicals > BT > Election_prac8

| Name | Date modified | Type | Size |
|---|---|---|---|
| .github | 21October2021 17:53 | File folder | |
| contracts | 21October2021 17:53 | File folder | |
| migrations | 21October2021 17:53 | File folder | |
| node_modules | 21October2021 17:55 | File folder | |
| src | 21October2021 17:53 | File folder | |
| test | 21October2021 17:53 | File folder | |
| .gitattributes | 27September2021 13:31 | Text Document | 1 KB |
| box-img-lg | 27September2021 13:31 | PNG File | 58 KB |
| box-img-sm | 27September2021 13:31 | PNG File | 8 KB |
| bs-config | 27September2021 13:31 | JSON File | 1 KB |
| LICENSE | 27September2021 13:31 | File | 2 KB |
| package | 27September2021 13:31 | JSON File | 1 KB |
| package-lock | 27September2021 13:31 | JSON File | 119 KB |
| truffle-config | 27September2021 13:31 | JavaScript File | 1 KB |

3. now go to contracts folder and you can see one migrations.sol file so you have to make same file as migrations.sol and name as **“Election.sol”** and then write below code into that file and save it.

**Election.sol**

```
pragma solidity 0.5.16;
contract Election {
   // Model a Candidate
   struct Candidate {
      uint id;
      string name;
      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
   );
   constructor () public {
      addCandidate("Candidate 1");
      addCandidate("Candidate 2");
   }
   function addCandidate (string memory _name) private {
```

```
            candidatesCount ++;
            candidates[candidatesCount] = Candidate(candidatesCount, _name, 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);
        }
    }
```

4. now go to migrations folder there you can see the file named as "1_initial_migration.js" same as above step you have to create a file named as **"2_deploy_contracts.js"** and add below code into that file and save it.

   **Note : the sol file name will be considered in this file so as we have created Election.sol so that the file will be called in this file.**
   **2_deploy_contracts.js**

   ```
   var Election = artifacts.require("./Election.sol");
   module.exports = function(deployer) {
     deployer.deploy(Election);
   };
   ```

5. so we are ready to create our frontend part of this application.
6. Go to src folder and see the index.html and you can create your own frontend but for reference the code of the frontend is below.

   **Index.html**

   ```
   <!DOCTYPE html>
   <html lang="en">
     <head>
       <meta charset="utf-8">
       <meta http-equiv="X-UA-Compatible" content="IE=edge">
       <meta name="viewport" content="width=device-width, initial-scale=1">
       <!-- The above 3 meta tags *must* come first in the head; any other head content
   ```

```
    must come *after* these tags -->
  <title>Election Results</title>

  <!-- Bootstrap -->
  <link href="css/bootstrap.min.css" rel="stylesheet">
  <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media
    queries -->
  <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
  <!--[if lt IE 9]>
    <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
    <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
  <![endif]-->
</head>
<body>
  <div class="container" style="width: 650px;">
    <div class="row">
      <div class="col-lg-12">
        <h1 class="text-center">Election Results</h1>
        <hr/>
        <br/>
        <div id="loader">
          <p class="text-center">Loading...</p>
        </div>
        <div id="content" style="display: none;">
          <table class="table">
            <thead>
              <tr>
                <th scope="col">#</th>
                <th scope="col">Name</th>
                <th scope="col">Votes</th>
              </tr>
            </thead>
            <tbody id="candidatesResults">
            </tbody>
          </table>
          <hr/>
          <form onSubmit="App.castVote(); return false;">
            <div class="form-group">
              <label for="candidatesSelect">Select Candidate</label>
              <select class="form-control" id="candidatesSelect">
              </select>
             </div>
             <button type="submit" class="btn btn-primary">Vote</button>
             <hr />
            </form>
            <p id="accountAddress" class="text-center"></p>
           </div>
          </div>
         </div>
        </div>
        <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
        <script
         src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
        <!-- Include all compiled plugins (below), or include individual files as needed -->
        <script src="js/bootstrap.min.js"></script>
        <script src="js/web3.min.js"></script>
        <script src="js/truffle-contract.js"></script>
        <script src="js/app.js"></script>
       </body>
      </html>
```

7. Now to connect metamask with our own created application we have to do some changes into the app.js file so go to src folder and then go to js folder and open js file then copy below code and paste it there.

**app.js**

```
App = {
  web3Provider: null,
  contracts: {},
  account: '0x0',
  hasVoted: false,
  init: function() {
    return App.initWeb3();
  },
  initWeb3: function() {
    // TODO: refactor conditional
    if (typeof web3 !== 'undefined') {
      // If a web3 instance is already provided by Meta Mask.
      App.web3Provider = web3.currentProvider;
      web3 = new Web3(web3.currentProvider);
    } else {
      // Specify default instance if no web3 instance provided
      App.web3Provider = new Web3.providers.HttpProvider('http://127.0.0.1:7545');
      web3 = new Web3(App.web3Provider);
    }
    return App.initContract();
  },
  initContract: function() {
    $.getJSON("Election.json", function(election) {
      // Instantiate a new truffle contract from the artifact
      App.contracts.Election = TruffleContract(election);
      // Connect provider to interact with contract
      App.contracts.Election.setProvider(App.web3Provider);
      App.listenForEvents();
      return App.render();
    });
  },
  // Listen for events emitted from the contract
  listenForEvents: function() {
    App.contracts.Election.deployed().then(function(instance) {
      // Restart Chrome if you are unable to receive this event
      // This is a known issue with Metamask
      // https://github.com/MetaMask/metamask-extension/issues/2393
      instance.votedEvent({}, {
        fromBlock: 0,
        toBlock: 'latest'
      }).watch(function(error, event) {
        console.log("event triggered", event)
        // Reload when a new vote is recorded
        App.render();
      });
    });
  },
  render: function() {
    var electionInstance;
    var loader = $("#loader");
    var content = $("#content");
    loader.show();
    content.hide();
    // Load account data
    web3.eth.getCoinbase(function(err, account) {
```

```
  if (err === null) {
    App.account = account;
    $("#accountAddress").html("Your Account: " + account);
  }
});
// Load contract data
App.contracts.Election.deployed().then(function(instance) {
  electionInstance = instance;
  return electionInstance.candidatesCount();
}).then(function(candidatesCount) {
  var candidatesResults = $("#candidatesResults");
  candidatesResults.empty();

  var candidatesSelect = $('#candidatesSelect');
  candidatesSelect.empty();

  for (var i = 1; i <= candidatesCount; i++) {
    electionInstance.candidates(i).then(function(candidate) {
      var id = candidate[0];
      var name = candidate[1];
      var voteCount = candidate[2];

      // Render candidate Result
      var candidateTemplate = "<tr><th>" + id + "</th><td>" + name + "</td><td>"
  + voteCount + "</td></tr>"
      candidatesResults.append(candidateTemplate);

      // Render candidate ballot option
      var candidateOption = "<option value='" + id + "' >" + name + "</ option>"
      candidatesSelect.append(candidateOption);
    });
  }
  return electionInstance.voters(App.account);
}).then(function(hasVoted) {
  // Do not allow a user to vote
  if(hasVoted) {
    $('form').hide();
  }
  loader.hide();
  content.show();
```

```
    }).catch(function(error) {
      console.warn(error);
    });
  },

  castVote: function() {
    var candidateId = $('#candidatesSelect').val();
    App.contracts.Election.deployed().then(function(instance) {
      return instance.vote(candidateId, { from: App.account });
    }).then(function(result) {
      // Wait for votes to update
      $("#content").hide();
      $("#loader").show();
    }).catch(function(err) {
      console.error(err);
    });
  }
};

$(function() {
  $(window).load(function() {
    App.init();
  });
});
```

8. So we are almost ready to launch the frontend, but lets first make one last file into the test folder named as **“election.js”** and write below code into that file.

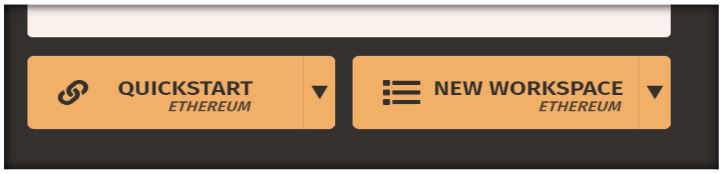
   **election.js**

```
var Election = artifacts.require("./Election.sol");

contract("Election", function(accounts) {
  var electionInstance;

  it("initializes with two candidates", function() {
    return Election.deployed().then(function(instance) {
      return instance.candidatesCount();
    }).then(function(count) {
      assert.equal(count, 2);
    });
  });
```

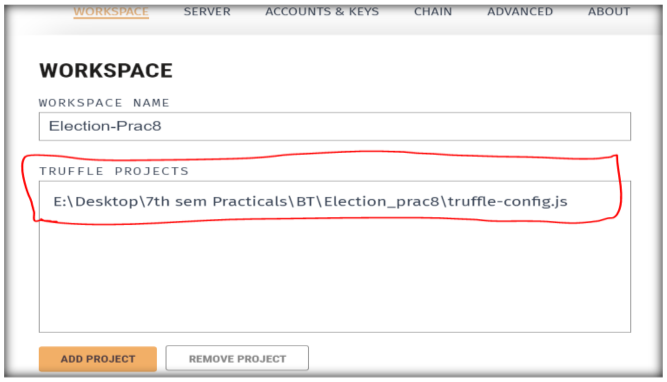
```
it("it initializes the candidates with the correct values", function() {
  return Election.deployed().then(function(instance) {
    electionInstance = instance;
    return electionInstance.candidates(1);
  }).then(function(candidate) {
    assert.equal(candidate[0], 1, "contains the correct id");
    assert.equal(candidate[1], "Candidate 1", "contains the correct name");
    assert.equal(candidate[2], 0, "contains the correct votes count");
    return electionInstance.candidates(2);
  }).then(function(candidate) {
    assert.equal(candidate[0], 2, "contains the correct id");
    assert.equal(candidate[1], "Candidate 2", "contains the correct name");
    assert.equal(candidate[2], 0, "contains the correct votes count");
  });
});

it("allows a voter to cast a vote", function() {
  return Election.deployed().then(function(instance) {
    electionInstance = instance;
    candidateId = 1;
    return electionInstance.vote(candidateId, { from: accounts[0] });
  }).then(function(receipt) {
    assert.equal(receipt.logs.length, 1, "an event was triggered");
    assert.equal(receipt.logs[0].event, "votedEvent", "the event type is correct");
    assert.equal(receipt.logs[0].args._candidateId.toNumber(),      candidateId,      "the
    candidate id is correct");
    return electionInstance.voters(accounts[0]);
  }).then(function(voted) {
    assert(voted, "the voter was marked as voted");
    return electionInstance.candidates(candidateId);
  }).then(function(candidate) {
    var voteCount = candidate[2];
    assert.equal(voteCount, 1, "increments the candidate's vote count");
  })
});

it("throws an exception for invalid candiates", function() {
  return Election.deployed().then(function(instance) {
    electionInstance = instance;
```

```
      return electionInstance.vote(99, { from: accounts[1] })
    }).then(assert.fail).catch(function(error) {
      assert(error.message.indexOf('revert') >= 0, "error message must contain revert");
      return electionInstance.candidates(1);
    }).then(function(candidate1) {
      var voteCount = candidate1[2];
      assert.equal(voteCount, 1, "candidate 1 did not receive any votes");
      return electionInstance.candidates(2);
    }).then(function(candidate2) {
      var voteCount = candidate2[2];
      assert.equal(voteCount, 0, "candidate 2 did not receive any votes");
    });
  });

  it("throws an exception for double voting", function() {
    return Election.deployed().then(function(instance) {
      electionInstance = instance;
      candidateId = 2;
      electionInstance.vote(candidateId, { from: accounts[1] });
      return electionInstance.candidates(candidateId);
    }).then(function(candidate) {
      var voteCount = candidate[2];
      assert.equal(voteCount, 1, "accepts first vote");
      // Try to vote again
      return electionInstance.vote(candidateId, { from: accounts[1] });
    }).then(assert.fail).catch(function(error) {
      assert(error.message.indexOf('revert') >= 0, "error message must contain revert");
      return electionInstance.candidates(1);
    }).then(function(candidate1) {
      var voteCount = candidate1[2];
      assert.equal(voteCount, 1, "candidate 1 did not receive any votes");
      return electionInstance.candidates(2);
    }).then(function(candidate2) {
      var voteCount = candidate2[2];
      assert.equal(voteCount, 1, "candidate 2 did not receive any votes");
    });
  });
});
```

9. Our part is almost done so open ganache you will get 2 options like below image.



10. Choose new workspace and name that you want. Then click on add project and choose **"truffle-config.js"** and then click on save workspace.

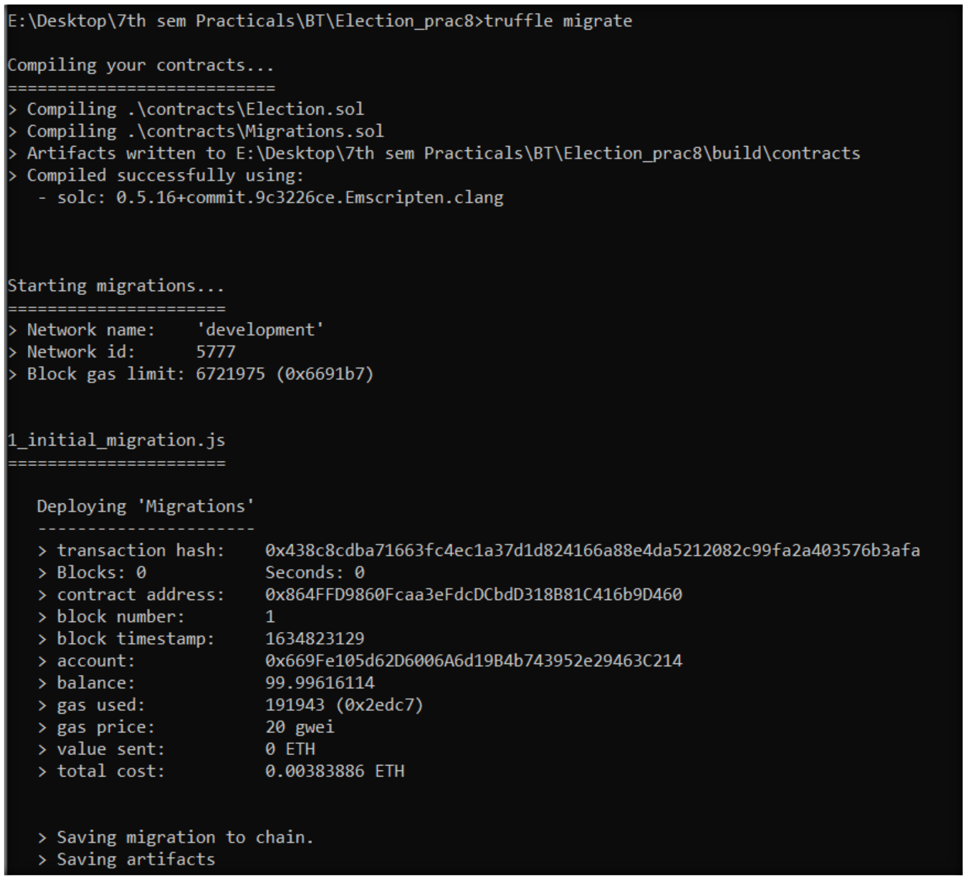


WORKSPACE

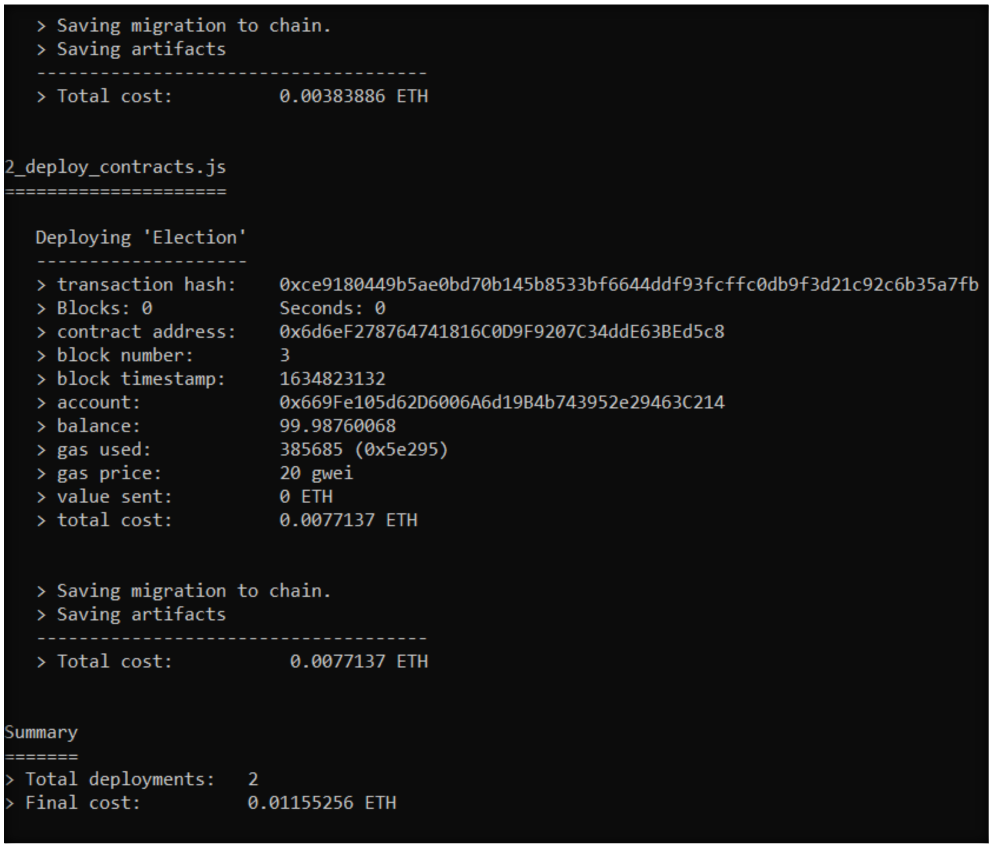
WORKSPACE NAME: Election-Prac8

TRUFFLE PROJECTS: E:\Desktop\7th sem Practicals\BT\Election_prac8\truffle-config.js

11. Now open cmd and run below command in the project folder.
    a. truffle migrate

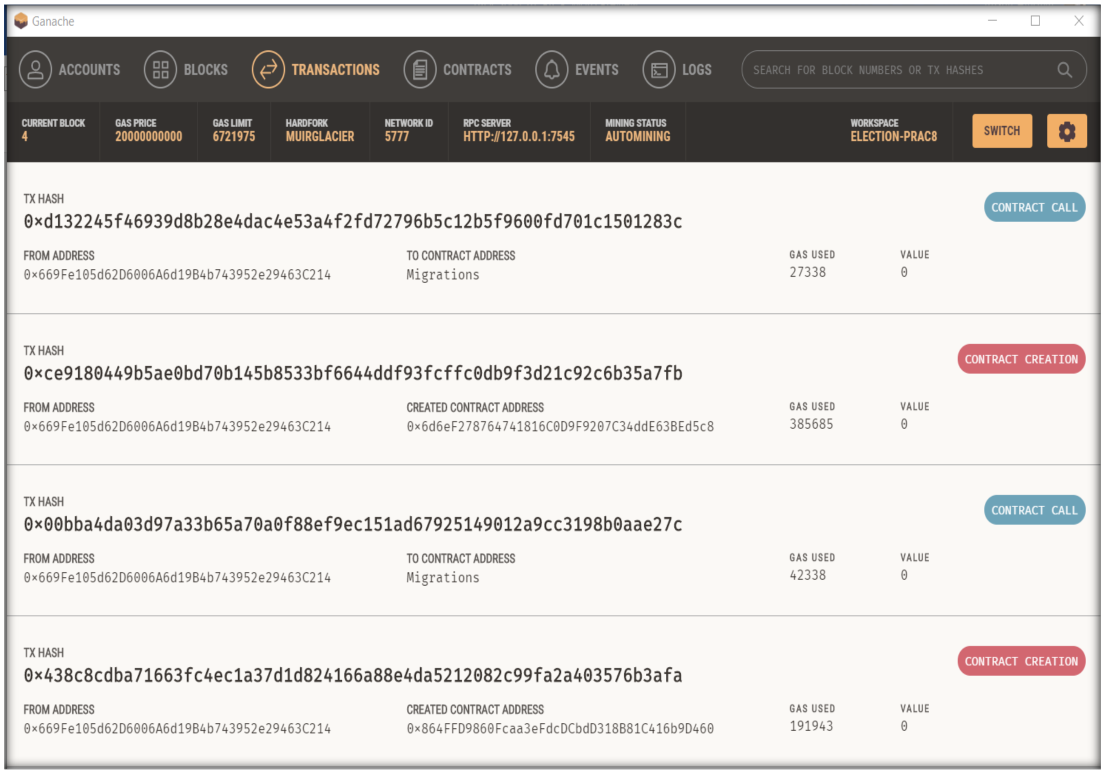


```
E:\Desktop\7th sem Practicals\BT\Election_prac8>truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\Election.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to E:\Desktop\7th sem Practicals\BT\Election_prac8\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang



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


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

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0x438c8cdba71663fc4ec1a37d1d824166a88e4da5212082c99fa2a403576b3afa
   > Blocks: 0            Seconds: 0
   > contract address:    0x864FFD9860Fcaa3eFdcDCbdD318B81C416b9D460
   > block number:        1
   > block timestamp:     1634823129
   > account:             0x669Fe105d62D6006A6d19B4b743952e29463C214
   > balance:             99.99616114
   > gas used:            191943 (0x2edc7)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00383886 ETH


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



```
   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00383886 ETH


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

   Deploying 'Election'
   --------------------
   > transaction hash:    0xce9180449b5ae0bd70b145b8533bf6644ddf93fcffc0db9f3d21c92c6b35a7fb
   > Blocks: 0            Seconds: 0
   > contract address:    0x6d6eF278764741816C0D9F9207C34ddE63BEd5c8
   > block number:        3
   > block timestamp:     1634823132
   > account:             0x669Fe105d62D6006A6d19B4b743952e29463C214
   > balance:             99.98760068
   > gas used:            385685 (0x5e295)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0077137 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:           0.0077137 ETH


Summary
=======
> Total deployments:   2
> Final cost:          0.01155256 ETH
```

12. you can see the transaction into the transactions panel of ganache and you can also see the contract that you have deployed just some seconds ago.



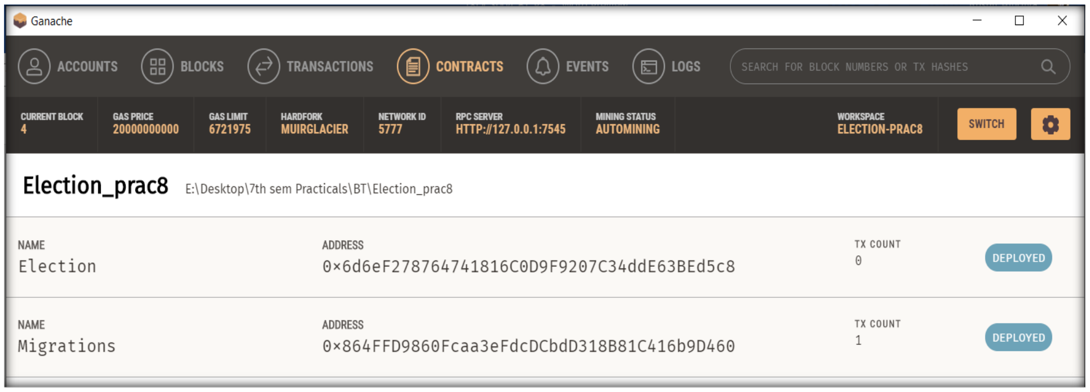
CURRENT BLOCK 4 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE ELECTION-PRAC8

TX HASH
0x d132245f46939d8b28e4dac4e53a4f2fd72796b5c12b5f9600fd701c1501283c
CONTRACT CALL
FROM ADDRESS 0x669Fe105d62D6006A6d19B4b743952e29463C214
TO CONTRACT ADDRESS Migrations
GAS USED 27338
VALUE 0

TX HASH
0xce9180449b5ae0bd70b145b8533bf6644ddf93fcffc0db9f3d21c92c6b35a7fb
CONTRACT CREATION
FROM ADDRESS 0x669Fe105d62D6006A6d19B4b743952e29463C214
CREATED CONTRACT ADDRESS 0x6d6eF278764741816C0D9F9207C34ddE63BEd5c8
GAS USED 385685
VALUE 0

TX HASH
0x00bba4da03d97a33b65a70a0f88ef9ec151ad67925149012a9cc3198b0aae27c
CONTRACT CALL
FROM ADDRESS 0x669Fe105d62D6006A6d19B4b743952e29463C214
TO CONTRACT ADDRESS Migrations
GAS USED 42338
VALUE 0

TX HASH
0x438c8cdba71663fc4ec1a37d1d824166a88e4da5212082c99fa2a403576b3afa
CONTRACT CREATION
FROM ADDRESS 0x669Fe105d62D6006A6d19B4b743952e29463C214
CREATED CONTRACT ADDRESS 0x864FFD9860Fcaa3eFdcDCbdD318B81C416b9D460
GAS USED 191943
VALUE 0

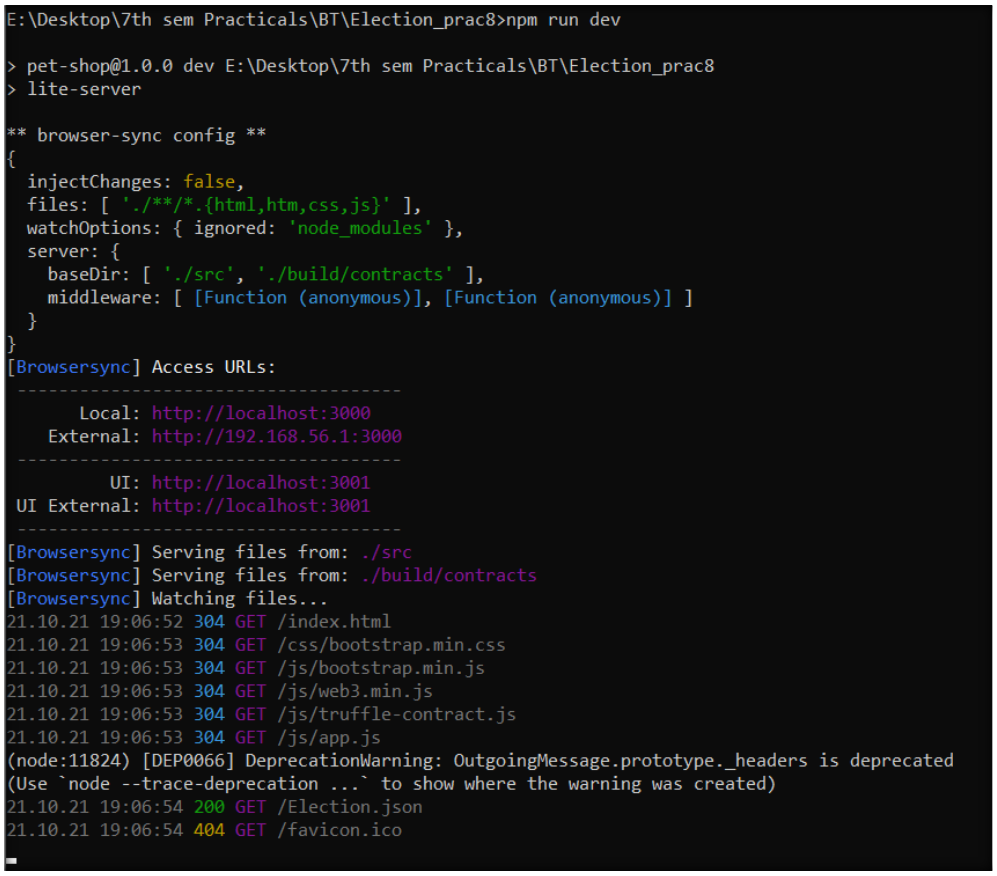


| CURRENT BLOCK | GAS PRICE | GAS LIMIT | HARDFORK | NETWORK ID | RPC SERVER | MINING STATUS | WORKSPACE |
|---|---|---|---|---|---|---|---|
| 4 | 20000000000 | 6721975 | MUIRGLACIER | 5777 | HTTP://127.0.0.1:7545 | AUTOMINING | ELECTION-PRAC8 |

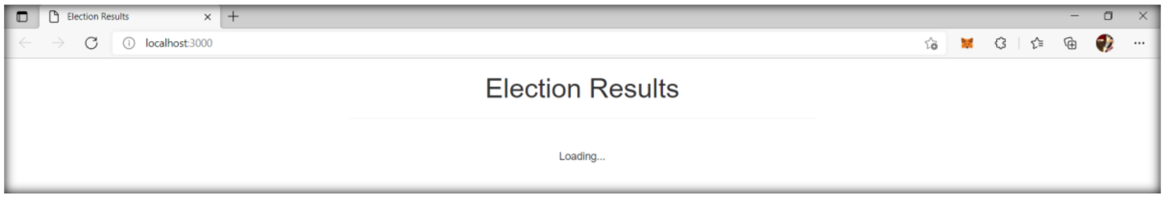
Election_prac8 E:\Desktop\7th sem Practicals\BT\Election_prac8

| NAME | ADDRESS | TX COUNT | |
|---|---|---|---|
| Election | 0x6d6eF278764741816C0D9F9207C34ddE63BEd5c8 | 0 | DEPLOYED |
| Migrations | 0x864FFD9860Fcaa3eFdcDCbdD318B81C416b9D460 | 1 | DEPLOYED |

13. run below command to see the frontend part into your localhost browser.
    a. npm run dev



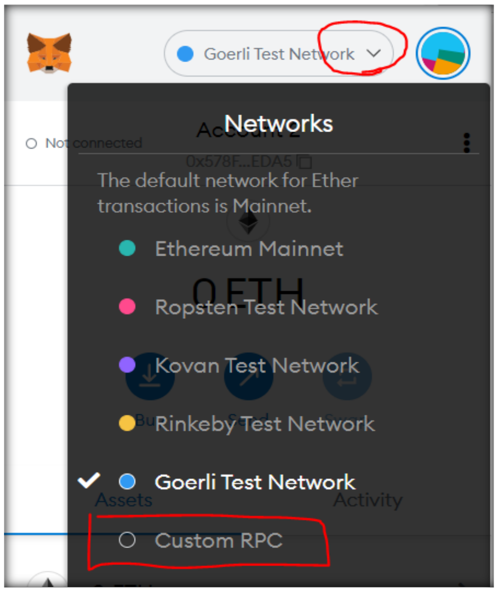
```
E:\Desktop\7th sem Practicals\BT\Election_prac8>npm run dev

> pet-shop@1.0.0 dev E:\Desktop\7th sem Practicals\BT\Election_prac8
> lite-server

** browser-sync config **
{
  injectChanges: false,
  files: [ './**/*.{html,htm,css,js}' ],
  watchOptions: { ignored: 'node_modules' },
  server: {
    baseDir: [ './src', './build/contracts' ],
    middleware: [ [Function (anonymous)], [Function (anonymous)] ]
  }
}
[Browsersync] Access URLs:
 ------------------------------------
       Local: http://localhost:3000
    External: http://192.168.56.1:3000
 ------------------------------------
          UI: http://localhost:3001
 UI External: http://localhost:3001
 ------------------------------------
[Browsersync] Serving files from: ./src
[Browsersync] Serving files from: ./build/contracts
[Browsersync] Watching files...
21.10.21 19:06:52 304 GET /index.html
21.10.21 19:06:53 304 GET /css/bootstrap.min.css
21.10.21 19:06:53 304 GET /js/bootstrap.min.js
21.10.21 19:06:53 304 GET /js/web3.min.js
21.10.21 19:06:53 304 GET /js/truffle-contract.js
21.10.21 19:06:53 304 GET /js/app.js
(node:11824) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is deprecated
(Use `node --trace-deprecation ...` to show where the warning was created)
21.10.21 19:06:54 200 GET /Election.json
21.10.21 19:06:54 404 GET /favicon.ico
```



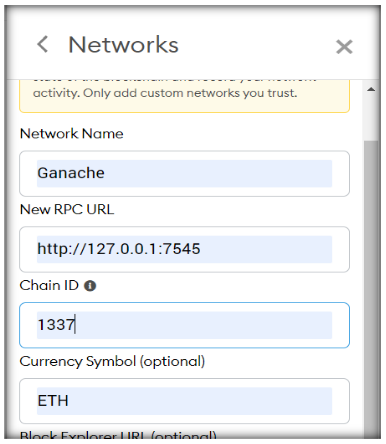
Election Results

Loading...

14. Now we have to connect the metamask with our application. Open metamask from extension panel in your browser then go to custom rpc in the network section.



15. Now fill the details from below image and click on save button.


Networks

activity. Only add custom networks you trust.

Network Name: Ganache

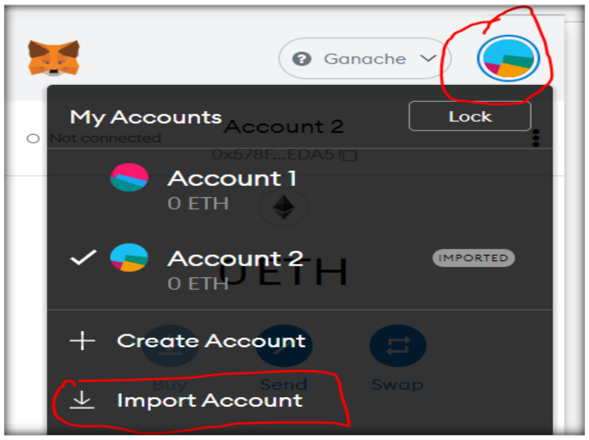
New RPC URL: http://127.0.0.1:7545

Chain ID: 1337

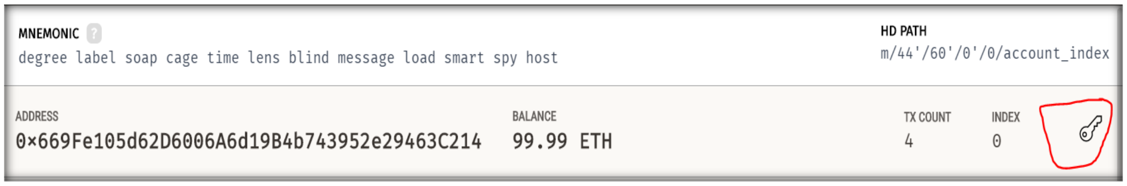
Currency Symbol (optional): ETH

Block Explorer URL (optional)

16. Go to account section and the click on import account.



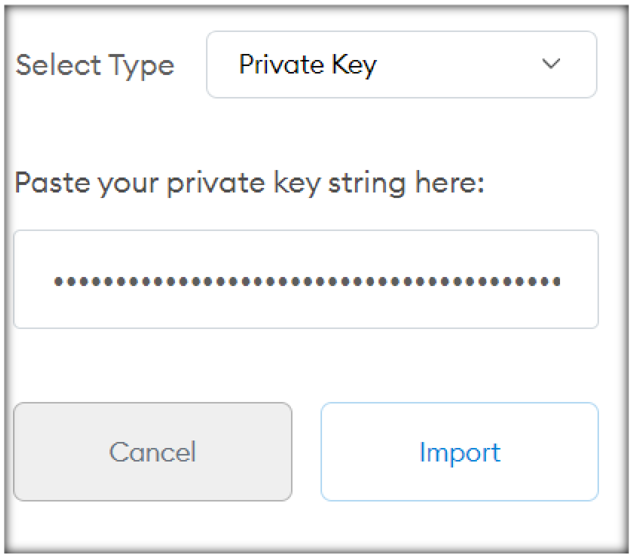
17. you can get the private key from the ganache account panel you can see the steps in below images. Then copy the private key and paste it into the metamask and click on import.



MNEMONIC
degree label soap cage time lens blind message load smart spy host

HD PATH
m/44'/60'/0'/0/account_index

| ADDRESS | BALANCE | TX COUNT | INDEX |
|---|---|---|---|
| 0x669Fe105d62D6006A6d19B4b743952e29463C214 | 99.99 ETH | 4 | 0 |

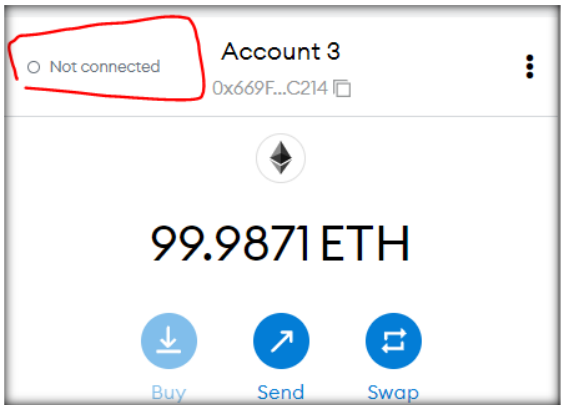


Select Type: Private Key

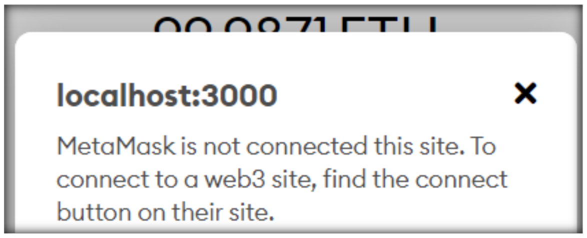
Paste your private key string here:

••••••••••••••••••••••••••••••••••••••••••••

18. click on **“not connected”** to connect to the localhost.



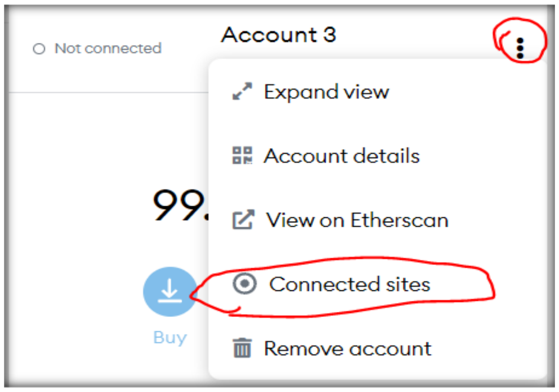
19. if you get an error like below image then follow step 20.

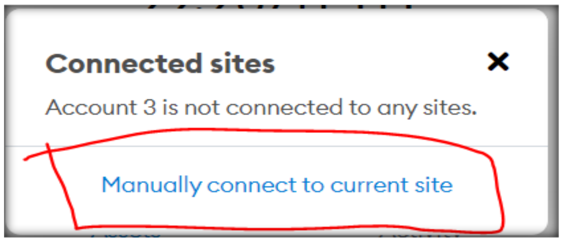


localhost:3000

MetaMask is not connected this site. To connect to a web3 site, find the connect button on their site.

20. click on the three dots and the click on connected sites. Then click on manually connect to the current site. Select account and click on next and then click on connect.

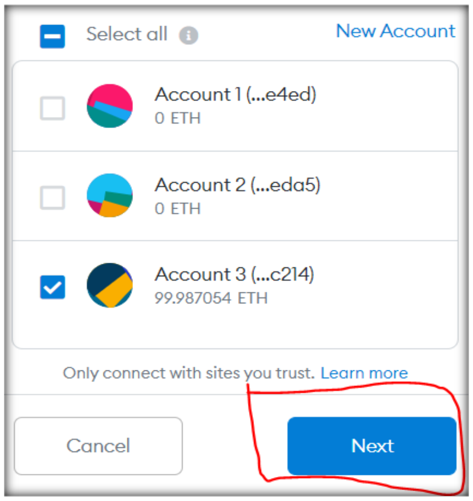




Connected sites

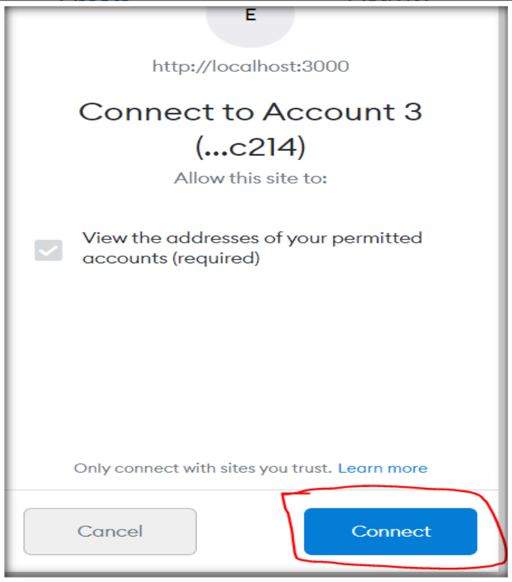
Account 3 is not connected to any sites.

Manually connect to current site



| | |
|---|---|
| ☐ | Account 1 (...e4ed) 0 ETH |
| ☐ | Account 2 (...eda5) 0 ETH |
| ✓ | Account 3 (...c214) 99.987054 ETH |

Only connect with sites you trust. Learn more



http://localhost:3000

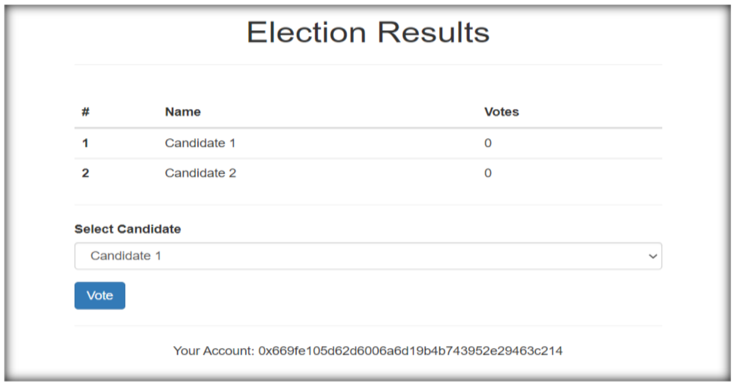
Connect to Account 3 (...c214)

Allow this site to:

✓ View the addresses of your permitted accounts (required)

Only connect with sites you trust. Learn more

21. after connecting to the metamask you can see the frontend like below image.



Election Results

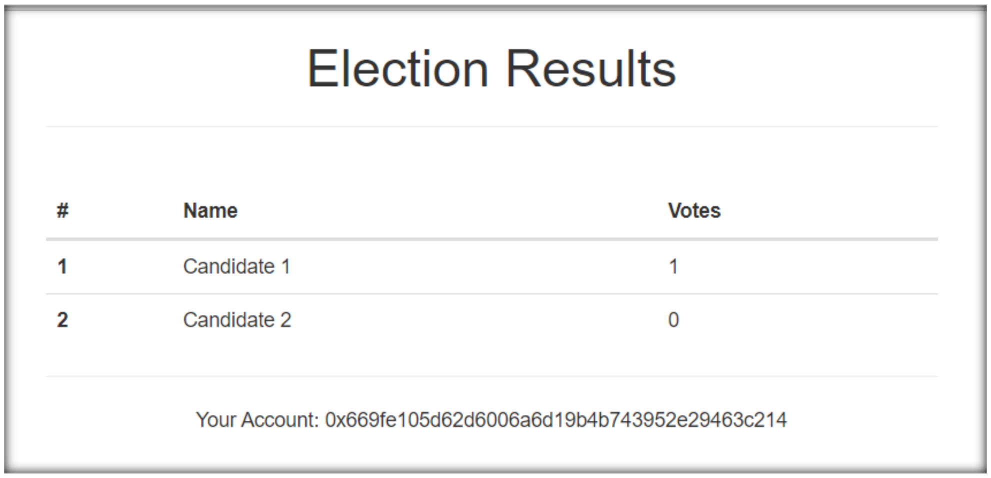
| # | Name | Votes |
|---|---|---|
| 1 | Candidate 1 | 0 |
| 2 | Candidate 2 | 0 |

Select Candidate

Your Account: 0x669fe105d62d6006a6d19b4b743952e29463c214

22. you can vote any candidate by choosing from drop down menu and by clicking on vote button. You can also change account and then you can vote from second account.

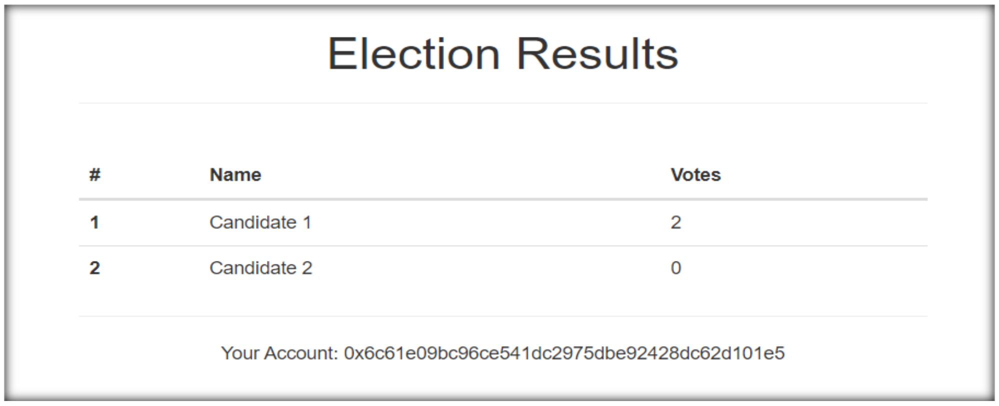
## OUTPUT:



Election Results

| # | Name | Votes |
|---|---|---|
| 1 | Candidate 1 | 1 |
| 2 | Candidate 2 | 0 |

Your Account: 0x669fe105d62d6006a6d19b4b743952e29463c214



Election Results

| # | Name | Votes |
|---|---|---|
| 1 | Candidate 1 | 2 |
| 2 | Candidate 2 | 0 |

Your Account: 0x6c61e09bc96ce541dc2975dbe92428dc62d101e5

## CONCLUSION:

In this practical, we thoroughly understood the concept of ganache, Truffle and implemented basic election system using it.