Assignment 9 Report by Chinmay Rai

STEP 0:
We connected to the private network by initializing a new node using the genesis file provided in assignment. There after we used the command "admin.addPeer("enode://<enode_number>@141.223.82.142:30303?discport=0")" , which began the synchronization of blockchain which was about 244158 blocks long.

Thereafter we set up the latest version of nodejs, npm and web3 on our computer.
Since we only need to make and UI for a simple wallet app, hence we don't need any smart contract. The web3.js librabry will suffice for our purpose.

STEP 1:
The code of the ethereum wallet is as follows :

==========

```
<!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>Chinmay's Ethereum Wallet</title>

        <!-- Bootstrap -->
          <link href="css/bootstrap.min.css" rel="stylesheet">

  </head>
  <body>

    <!-- The code between the following div tags defines the structure of web page and defines elements like buttons, tables etc. -->
        <div class="container" style="width: 650px;">
      <div class="row">
        <div class="col-lg-12">
```

```
 <!-- Heading -->
        <h1 class="text-center">Chinmay's Ethereum Wallet</h1>
        <hr/>
        <br/>

        <div id="loader">

          <!-- Sub-heading -->
          <p class="text-center">List of Accounts</p>

          <!-- when Update button is clicked, it calls update() -->
          <p class="text-center"> <button type="button" align="middle"
onClick="update();">Update</button></p>

        </div>

        <!-- Contents (Two tables) -->
        <div id="content">

          <!-- Table 1 -->
          <table class="table">
            <thead>
              <tr>
                <!-- Table 1 headers -->
                <th scope="col">#</th>
                <th scope="col">Address</th>
                <th scope="col">Balance (in Ether)</th>
              </tr>
            </thead>
              <!-- row 1 -->
              <tr>
                <td> 1 </td>
                <td> <div id="Account1"></div> </td>
                <td> <div id="Balance1"></div> </td>
              </tr>

              <!-- row 2 -->
              <tr>
                <td> 2 </td>
                <td> <div id="Account2"></div> </td>
                <td> <div id="Balance2"></div> </td>
              </tr>
```

```
    <!-- row 3 -->
    <tr>
      <td> 3 </td>
      <td> <div id="Account3"></div> </td>
      <td> <div id="Balance3"></div> </td>
    </tr>

    <!-- row 4 -->
    <tr>
      <td> 4 </td>
      <td> <div id="Account4"></div> </td>
      <td> <div id="Balance4"></div> </td>
    </tr>

    <!-- last row -->
    <tr>
      <td></td>
      <td> Total</td>
      <td> <div id="Total_Balance"></div> </td>
    </tr>
  </tbody>
</table>

<!-- Table 2 -->
<table class="table">

  <!-- row for selecting the Transmitter -->
  <tr>
    <td>Transmitter</td>
    <td>
    <input list="browsers" name="sender_address">
      <datalist id="browsers">
        <option value="0x9a638026e514f4a639571e2c0d8a37d1cb316fc4">
        <option value="0x90a7baed4aafd620a0425e44ba0affbf27300c3b">
        <option value="0x82b16786f38aedfdaf61879d36f64e1e6505d475">
        <option value="0x8f922442f3ecaf5feade3abea1d51303fd212594">
      </datalist>
    </td>
  </tr>
```

```
<!-- row for selecting the rreceiver -->
          <tr>
            <td>Recipient </td>
            <td><input type="text" name="recipient_address"><br></td>
          </tr>

          <!-- row for Amount to transfer -->
          <tr>
            <td>Value </td>
            <td><input type="number" name="value_to_transfer"><br></td>
          </tr>

          <!-- row for password -->
          <tr>
            <td>Password </td>
            <td><input type="password" name="password"><br></td>
            <td><button type="button" onclick="send_transaction();">Send</button></td>
          </tr>
        </table>


        <!-- Transaction Hash of last transaction -->
        <label for="output">Hash of last transaction </label><div id="output"></div>

        <hr/>
      </div>
    </div>
  </div>
</div>




  <!-- Including Important JavaScirpt Libraries -->
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
  <script src="js/bootstrap.min.js"></script>
  <script src="js/web3.min.js"></script>
```

```
  <!-- The code between the following <script> tags establishes our connection with the
blockchain using web3.js -->
    <script>

      //establishing the connection
      if (typeof web3 !== 'undefined') {
        web3 = new Web3(web3.currentProvider);
      } else {
        web3 = new Web3(new Web3.providers.HttpProvider('http://localhost:8545'));
      }

      //get the acccount addresses
      document.getElementById('Account1').innerText = web3.eth.accounts[0];
      document.getElementById('Account2').innerText = web3.eth.accounts[1];
      document.getElementById('Account3').innerText = web3.eth.accounts[2];
      document.getElementById('Account4').innerText = web3.eth.accounts[3];

      //get the acccount balances in ethers
      document.getElementById('Balance1').innerText =
web3.fromWei(web3.eth.getBalance(web3.eth.accounts[0]));
      document.getElementById('Balance2').innerText =
web3.fromWei(web3.eth.getBalance(web3.eth.accounts[1]));
      document.getElementById('Balance3').innerText =
web3.fromWei(web3.eth.getBalance(web3.eth.accounts[2]));
      document.getElementById('Balance4').innerText =
web3.fromWei(web3.eth.getBalance(web3.eth.accounts[3]));


      //function for updating the acccount balances
      function update() {
        var bal1 = web3.fromWei(web3.eth.getBalance(web3.eth.accounts[0]));
        var bal2 = web3.fromWei(web3.eth.getBalance(web3.eth.accounts[1]));
        var bal3 = web3.fromWei(web3.eth.getBalance(web3.eth.accounts[2]));
        var bal4 = web3.fromWei(web3.eth.getBalance(web3.eth.accounts[3]));

        document.getElementById('Balance1').innerText = bal1;
        document.getElementById('Balance2').innerText = bal2;
        document.getElementById('Balance3').innerText = bal3;
        document.getElementById('Balance4').innerText = bal4;
        document.getElementById('Total_Balance').innerText = parseInt(bal1) + parseInt(bal2) +
parseInt(bal3) +parseInt(bal4);

      }
```

```
    //function making transactions
    function send_transaction() {

      //reading Sender's & reciever's address, Amount, password
      var sen = document.getElementsByName("sender_address")[0].value;
      var rec = document.getElementsByName("recipient_address")[0].value;
      var amount = parseInt(document.getElementsByName("value_to_transfer")[0].value,10) *
1000000000000000000;
      var pwd = document.getElementsByName("password")[0].value;

      //unlocking the sender's account with password
      web3.personal.unlockAccount(sen,pwd);

      //executing transaction and outputting hash of transaction
      web3.eth.sendTransaction({ from: sen, to: rec, value: amount }, function(error, hash){
        if(error){document.getElementById('output').innerText = error;}
        else document.getElementById('output').innerText = hash ;
       });

    }


  </script>
 </body>
</html>
```

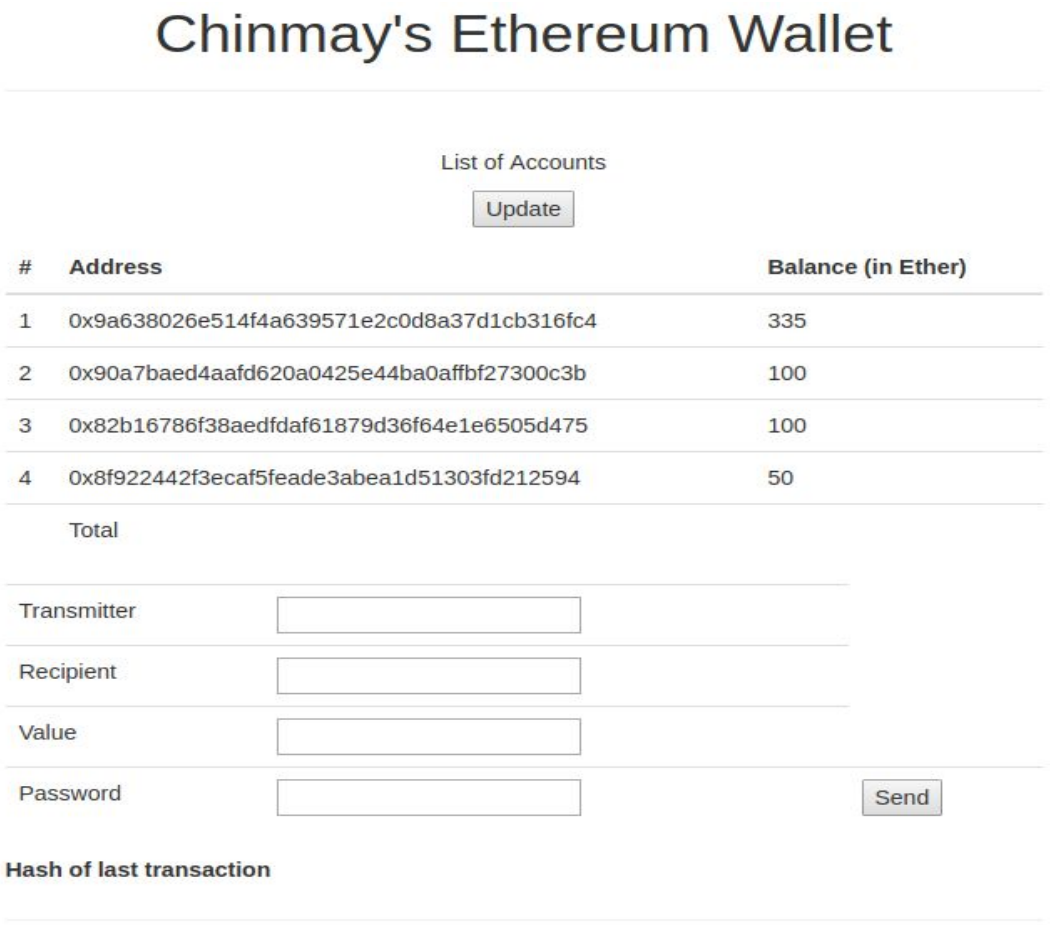
==========

The code is essentially divided into two parts. The HTML part defines the layout and elements of user Interface, whereas the JavaScript part define the behaviour of those elements and help us interact with the blockchain using web3.js.

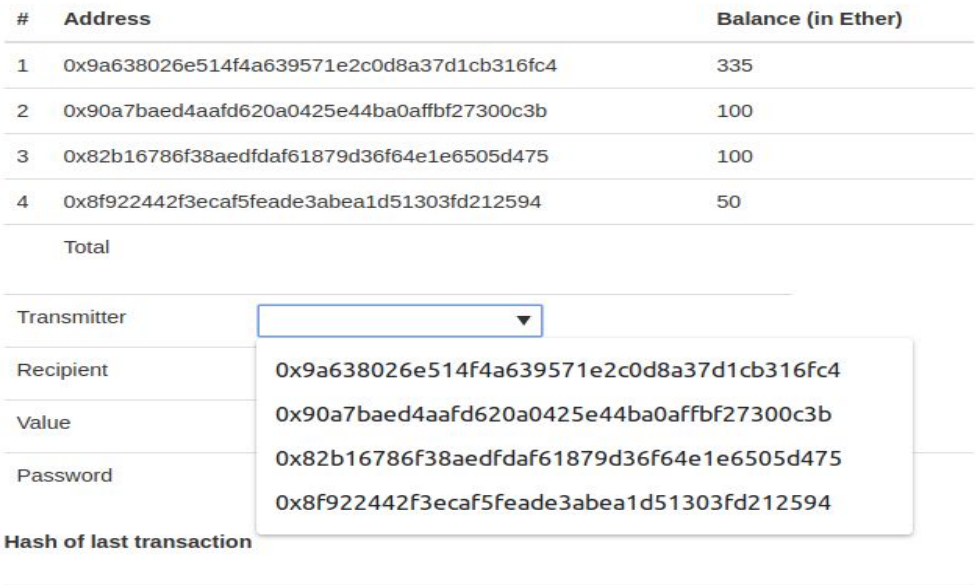
There are two main funtions, one for updating the balances and other for making transactions. The detailed description of code is made using comments.

STEP2:

The Image of UI is as follows:

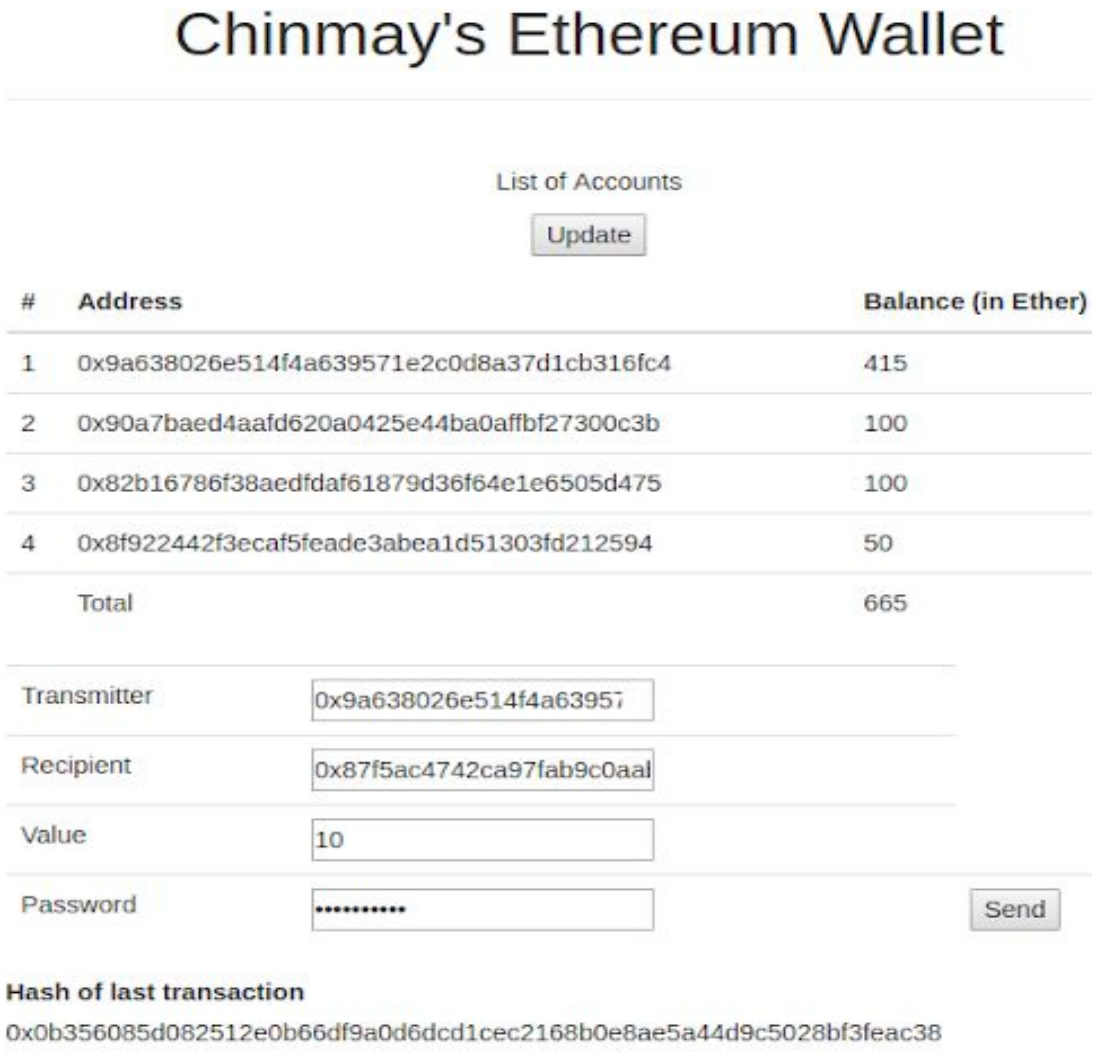


The Transmitter Box also shows all the accounts which are available to give ethers:



STEP 3:

10 ethers are sent to address 0x87f5ac4742ca97fab9c0aab50aec11beceaff24d. Following is the screen after transaction.



STEP 4:
We check the transaction using the hash:
0x0b356085d082512e0b66df9a0d6dcd1cec2168b0e8ae5a44d9c5028bf3feac38
Output:

{
 blockHash: "0xe18cf725c540ec9bb06870249b9ab1bf3fd9646c2ddfea38b90a4871690db00b",
 blockNumber: 244293,
 from: "0x9a638026e514f4a639571e2c0d8a37d1cb316fc4",
 gas: 90000,
 gasPrice: 1000000000,
 hash: "0x0b356085d082512e0b66df9a0d6dcd1cec2168b0e8ae5a44d9c5028bf3feac38",

input: "0x",
nonce: 6,
r: "0x7f851a009dc791305bfba72be21da6b833813830a94bdb3ab3cbb94f403388af",
s: "0x6d8adb1072b811a70d4b2e0fa698bf5ce224cf45c196c7b787db7cc04b2c6f58",
to: "0x87f5ac4742ca97fab9c0aab50aec11beceaff24d",
transactionIndex: 0,
v: "0x3f2f",
value: 100000000000000000000
}

STEP 5 :

Checking the block number 244293, output:

{
difficulty: 2072265,
extraData: "0xd883010811846765746888676f312e31302e31856c696e7578",
gasLimit: 5375900,
gasUsed: 21000,
hash: "0xe18cf725c540ec9bb06870249b9ab1bf3fd9646c2ddfea38b90a4871690db00b",
logsBloom:
"0x000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000",
miner: "0x9a638026e514f4a639571e2c0d8a37d1cb316fc4",
mixHash: "0x7127a93175108404776f03e9cdfbbb47387467421f2bc9ea82e57d9d7c003b37",
nonce: "0x36c8d4a9b92a0023",
number: 244293,
parentHash:
"0xdaea171b1726b017ba9bebc23fd0b9c4decd8dba51e91ed14ab671948c7895a0",
receiptsRoot:
"0xbcf3ed5fa7e7fae0a67257853e7acb3bf8177a01a8b5d5508ad9812d96121afd",
sha3Uncles:
"0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
size: 654,
stateRoot: "0x15ba0c4ec516cbd594015860864fce996036a598457503f4ef1dbbba8d038e86",
timestamp: 1541753541,
totalDifficulty: 881882046531,

```
  transactions:
["0x0b356085d082512e0b66df9a0d6dcd1cec2168b0e8ae5a44d9c5028bf3feac38"],
  transactionsRoot:
"0x7223dbd73338b2837a4daf52b09b9fd51dc47a9678d75d3e80a7aceb89fa23db",
  uncles: []
}
```

We can see that it has only one transaction which is the same as the one our system returned.

Another thing which is needed to be kept in mind while executing the code is that the directory sturucture for the code dependencies should be as follows.

```
.
├── css
│   ├── bootstrap.min.css
│   └── bootstrap.min.css.map
├── fonts
│   ├── glyphicons-halflings-regular.eot
│   ├── glyphicons-halflings-regular.svg
│   ├── glyphicons-halflings-regular.ttf
│   ├── glyphicons-halflings-regular.woff
│   └── glyphicons-halflings-regular.woff2
├── index.html
└── js
    ├── bootstrap.min.js
    └── web3.min.js

3 directories, 10 files
```