E-Voting system using Blockchain

Team Members

Hari	- CS19BTECH11039
Sreehitha	- CS19BTECH11016
Hemanth	- ES19BTECH11003
Venkata Surya	- CS19BTECH11014
Nisha	- CS19BTECH11012
Naveen	- CS19BTECH11009

Introduction

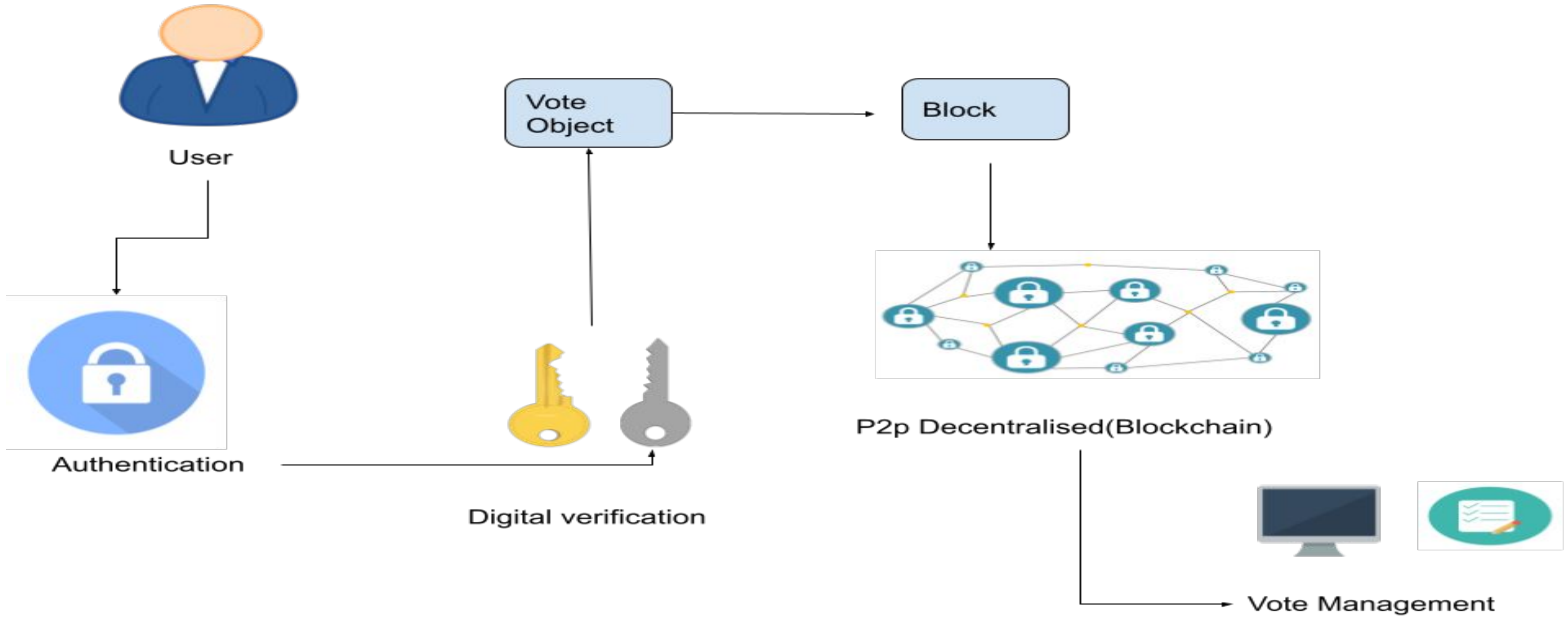
The main objective of this project is to deliver more convenient and secure applications through the use of blockchain technology implemented via decentralized p2p network connection.

The anonymities of voters, the security of ballot transmission and the verifiability of votes during the billing phase are the most fundamental requirements for voting.

The anonymity and security can be achieved by the secret sharing scheme with public and private key cryptosystem while the verifiability of votes can be realized by taking advantage of the transparency and non-repudiation of blockchain.

Voters can calculate the ballots and verify the election results on their own without a trusted third party.

System design

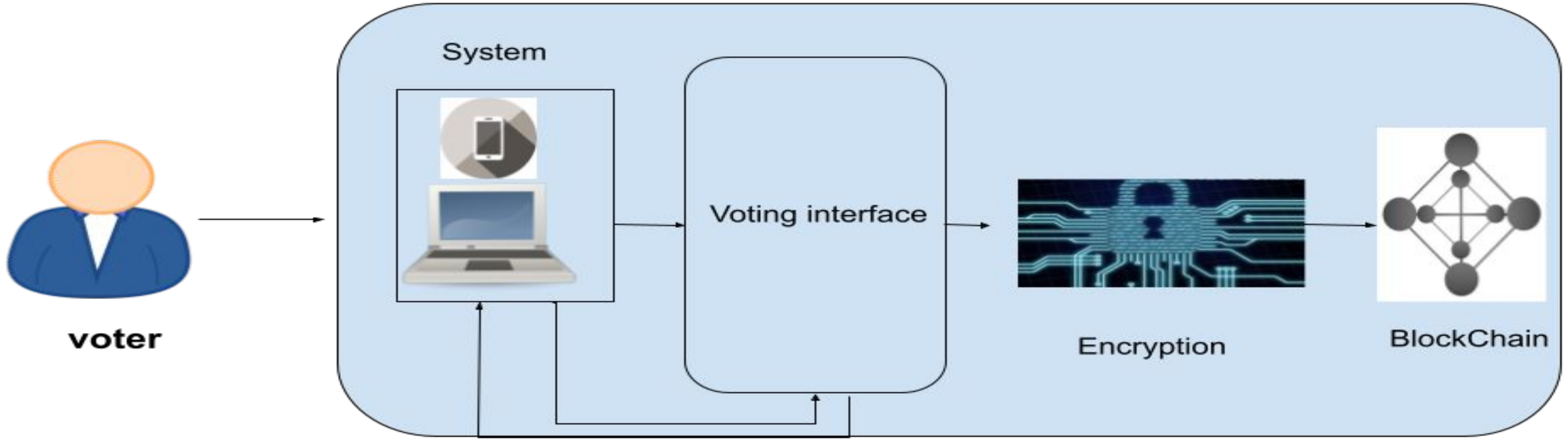


System Design Details

- The user logs in to the portal first, the portal will be developed using flask. The user needs to enter the private key provided by the system. Thus only valid users can vote in the system.
- The user then chooses an option to vote. The system then uses digital signature to verify the vote. This is achieved using Elliptical Curve - secp256k1 to generate the key pairs.
- Once the verification is done, a new vote object is created with encrypted entities and will be requested to add in blockchain.
- The blockchain already has the details of the voter(public key) in it. So the blockchain will check if the user has already voted and then proceeds accordingly.

- Thus a vote can be inscribed into the blockchain.
- Every node in the peer-peer network will have access to the blockchain and new vote will be added accordingly after being checked.
- Using the tcp connections which are the part of p2p network, peers in the network communicate with each other to know the status, validity of the chain. The requests are also made using these connections.
- This modelling is done taken into consideration of factors such as high-availability, data integrity, transparency, data immutability.
- Finally one can get the voting count as it will be included in the block. Anonymity is maintained as the votes are updated with the candidate’s public key and the count(user is different from candidate).

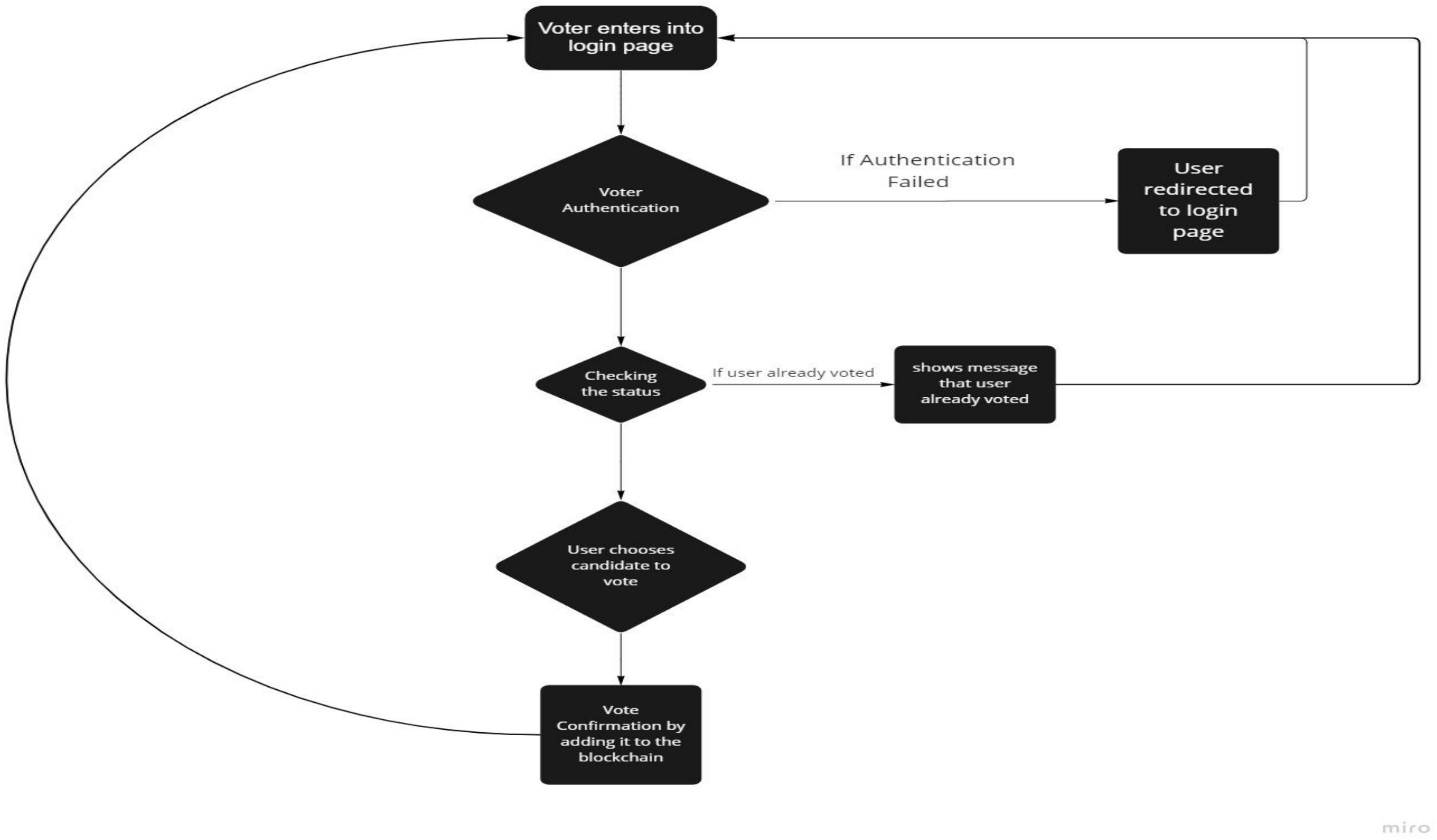
Block diagram



Block Diagram Details

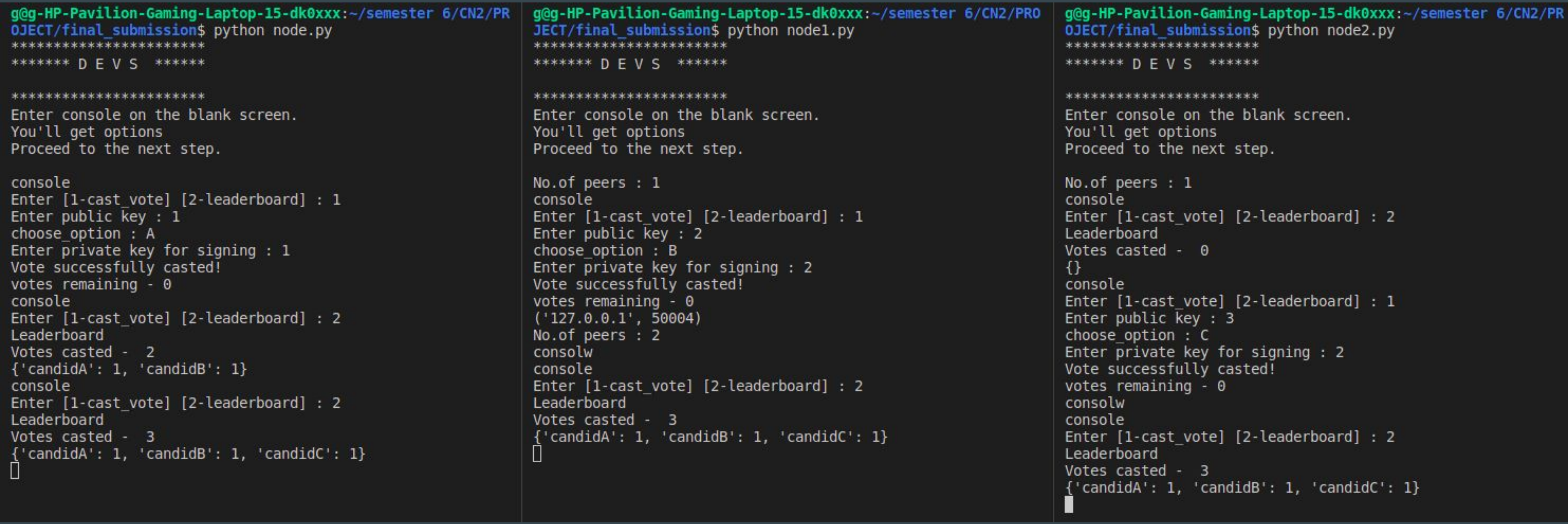
- In above block diagram we can basically see 4 blocks i.e, system, voter Interface, Encryption, Blockchain
- First voter is taken to the voter interface.
- Next the details are encrypted in to a block.
- Further the block is added into the blockchain.

Flow chart



Voter enters into login page
Voter Authentication
If Authentication Failed
User redirected to login page
Checking the status
If user already voted
shows message that user already voted
User chooses candidate to vote
Vote Confirmation by adding it to the blockchain
miro

Working of E - Voting System



```
g@g-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/semester 6/CN2/PR
OJECT/final_submission$ python node.py
***********************
******* D E V S  ******

***********************
Enter console on the blank screen.
You'll get options
Proceed to the next step.

console
Enter [1-cast_vote] [2-leaderboard] : 1
Enter public key : 1
choose_option : A
Enter private key for signing : 1
Vote successfully casted!
votes remaining - 0
console
Enter [1-cast_vote] [2-leaderboard] : 2
Leaderboard
Votes casted -  2
{'candidA': 1, 'candidB': 1}
console
Enter [1-cast_vote] [2-leaderboard] : 2
Leaderboard
Votes casted -  3
{'candidA': 1, 'candidB': 1, 'candidC': 1}
```

```
g@g-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/semester 6/CN2/PRO
JECT/final_submission$ python node1.py
***********************
******* D E V S  ******

***********************
Enter console on the blank screen.
You'll get options
Proceed to the next step.

No.of peers : 1
console
Enter [1-cast_vote] [2-leaderboard] : 1
Enter public key : 2
choose_option : B
Enter private key for signing : 2
Vote successfully casted!
votes remaining - 0
('127.0.0.1', 50004)
No.of peers : 2
consolw
console
Enter [1-cast_vote] [2-leaderboard] : 2
Leaderboard
Votes casted -  3
{'candidA': 1, 'candidB': 1, 'candidC': 1}
```

```
g@g-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/semester 6/CN2/PR
OJECT/final_submission$ python node2.py
***********************
******* D E V S  ******

***********************
Enter console on the blank screen.
You'll get options
Proceed to the next step.

No.of peers : 1
console
Enter [1-cast_vote] [2-leaderboard] : 2
Leaderboard
Votes casted -  0
{}
console
Enter [1-cast_vote] [2-leaderboard] : 1
Enter public key : 3
choose_option : C
Enter private key for signing : 2
Vote successfully casted!
votes remaining - 0
consolw
console
Enter [1-cast_vote] [2-leaderboard] : 2
Leaderboard
Votes casted -  3
{'candidA': 1, 'candidB': 1, 'candidC': 1}
```

The above fig. Show the clear working of the system. We can see that 3 users have casted their votes and we can also display the Leader board which is kept updated in every user's console

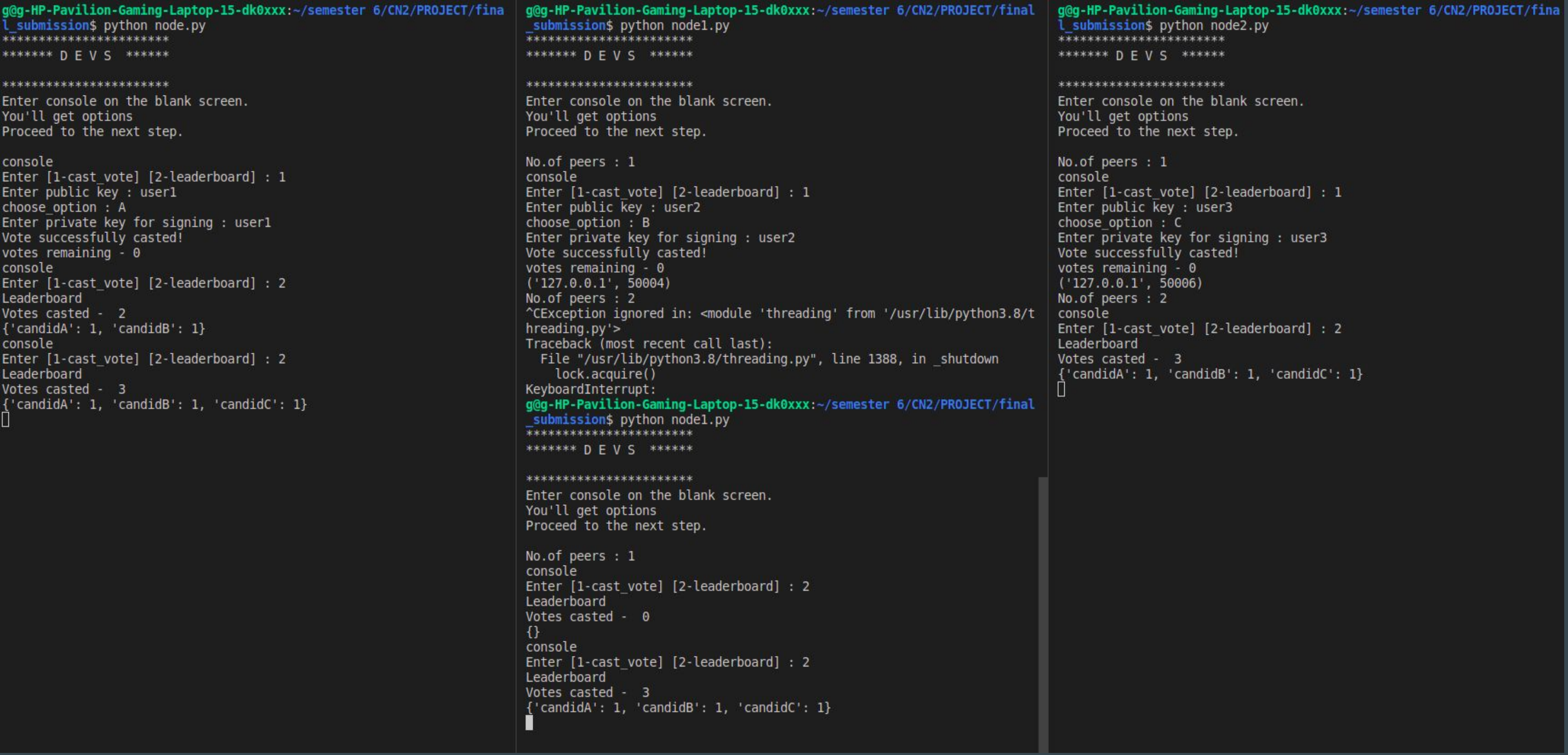
Key Point

High availability:

- Even though if the network is busy are few went to busy state, remaining servers can make the tasks done smoothly

Immutability:

- the ability for a blockchain ledger to remain a permanent, indelible, and unalterable history of transactions
- Suppose if a server/node gets tampered, then the block chain validation will be done according to the majority match in the remaining servers/nodes.



```
g@g-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/semester 6/CN2/PROJECT/fina
l_submission$ python node.py
**********************
******* D E V S  ******

**********************
Enter console on the blank screen.
You'll get options
Proceed to the next step.

console
Enter [1-cast_vote] [2-leaderboard] : 1
Enter public key : user1
choose_option : A
Enter private key for signing : user1
Vote successfully casted!
votes remaining - 0
console
Enter [1-cast_vote] [2-leaderboard] : 2
Leaderboard
Votes casted -  2
{'candidA': 1, 'candidB': 1}
console
Enter [1-cast_vote] [2-leaderboard] : 2
Leaderboard
Votes casted -  3
{'candidA': 1, 'candidB': 1, 'candidC': 1}
```

```
g@g-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/semester 6/CN2/PROJECT/final
_submission$ python node1.py
**********************
******* D E V S  ******

**********************
Enter console on the blank screen.
You'll get options
Proceed to the next step.

No.of peers : 1
console
Enter [1-cast_vote] [2-leaderboard] : 1
Enter public key : user2
choose_option : B
Enter private key for signing : user2
Vote successfully casted!
votes remaining - 0
('127.0.0.1', 50004)
No.of peers : 2
^CException ignored in: <module 'threading' from '/usr/lib/python3.8/t
hreading.py'>
Traceback (most recent call last):
  File "/usr/lib/python3.8/threading.py", line 1388, in _shutdown
    lock.acquire()
KeyboardInterrupt:
g@g-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/semester 6/CN2/PROJECT/final
_submission$ python node1.py
**********************
******* D E V S  ******

**********************
Enter console on the blank screen.
You'll get options
Proceed to the next step.

No.of peers : 1
console
Enter [1-cast_vote] [2-leaderboard] : 2
Leaderboard
Votes casted -  0
{}
console
Enter [1-cast_vote] [2-leaderboard] : 2
Leaderboard
Votes casted -  3
{'candidA': 1, 'candidB': 1, 'candidC': 1}
```

```
g@g-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/semester 6/CN2/PROJECT/fina
l_submission$ python node2.py
**********************
******* D E V S  ******

**********************
Enter console on the blank screen.
You'll get options
Proceed to the next step.

No.of peers : 1
console
Enter [1-cast_vote] [2-leaderboard] : 1
Enter public key : user3
choose_option : C
Enter private key for signing : user3
Vote successfully casted!
votes remaining - 0
('127.0.0.1', 50006)
No.of peers : 2
console
Enter [1-cast_vote] [2-leaderboard] : 2
Leaderboard
Votes casted -  3
{'candidA': 1, 'candidB': 1, 'candidC': 1}
```

High availability

In the above slide , First we see that user 1, user2 has casted their votes then server 2 is down(we close console2) thereafter user3 has casted his/her vote in absence of user2 but when user2 is again active we can see that user3 has casted some vote in Leaderboard

Transparency:

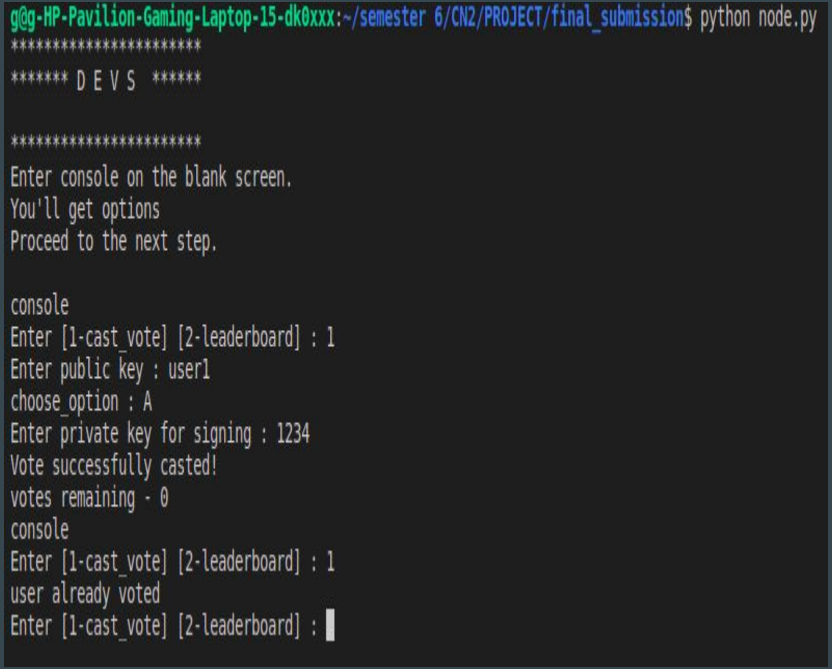
- We can check the status(whether he/she voted or not) of vote of a person,
  And can also get the final vote count

Decentralized:

- As we are using peer to peer network, we are maintaining decentralization.

Double voting

- We make sure no voter is allowed to vote twice.
- In the right fig. It is clearly seen that user1 has casted his/her vote and next tried to vote again but there is an message on console saying “user already voted”



```
g@g-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/semester 6/CN2/PROJECT/final_submission$ python node.py
**********************
******* D E V S  ******

**********************
Enter console on the blank screen.
You'll get options
Proceed to the next step.

console
Enter [1-cast_vote] [2-leaderboard] : 1
Enter public key : user1
choose_option : A
Enter private key for signing : 1234
Vote successfully casted!
votes remaining - 0
console
Enter [1-cast_vote] [2-leaderboard] : 1
user already voted
Enter [1-cast_vote] [2-leaderboard] : 
```

References

1. https://www.researchgate.net/publication/321947971_Decentralized_E-Voting_Systems_Based_on_the_Blockchain_Technology
2. https://ieeexplore.ieee.org/document/8944820
3. https://www.ijert.org/research/implementation-of-secure-voting-system-using-blockchain-IJERTV9IS060974.pdf

Read me

- The files node*.py represents a user.
- Blockchain.py and instructions.py are dependencies of node*.py
- Prefer running node*.py with python3
- A clean interface appears at the start of the program and guides user how to use the application.
- Must have dependencies : pickle, ecdsa
- The files blockchain.py and instructions.py are imported into nodes such that, user need not explicitly run those files.