Advance network System Programming Homework 5 report
Yifei Wang
cs6675

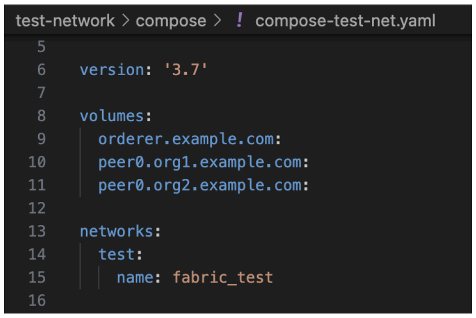
Learning Process:

I worked with hyperledger fabric and developed a simple voting web app using node js and express js. I started with learning about hyperledger fabric, its internal structure of how the client application, endorser,orderer, and committer works, and the workflow for when a transaction is being made. After getting a basic idea, I downloaded the fabric-samples repository(https://github.com/hyperledger/fabric-samples) and ran the most basic example of asset transfer. While reading, trying and running the code, I got familiar with fabric's node js API, so I modified the transfer-asset-basic sample and made my voting app.

Environment setup and app development:

1. Setup docker container and nodes

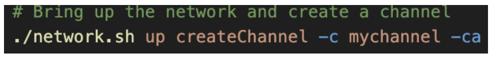
During the setup, I first deploy the network. This is when I can define my network structure, for example, how many endorser nodes, how many committer nodes, and how should the endorsement policy be. In my example setup command, I used the most basic setup, with 2 peers, 1 for endorser and 1 for committer(as shown in the screenshot below). The endorsement policy would be as long as the 1 endorser runs the code and gives an output, the output would be committed without proof check.



test-network > compose > ! compose-test-net.yaml

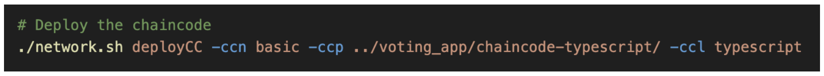
```
version: '3.7'

volumes:
  orderer.example.com:
  peer0.org1.example.com:
  peer0.org2.example.com:

networks:
  test:
    name: fabric_test
```

I would run the below commands to set up the docker environment with the defined number of nodes. This command brings up the network, and creates channels.



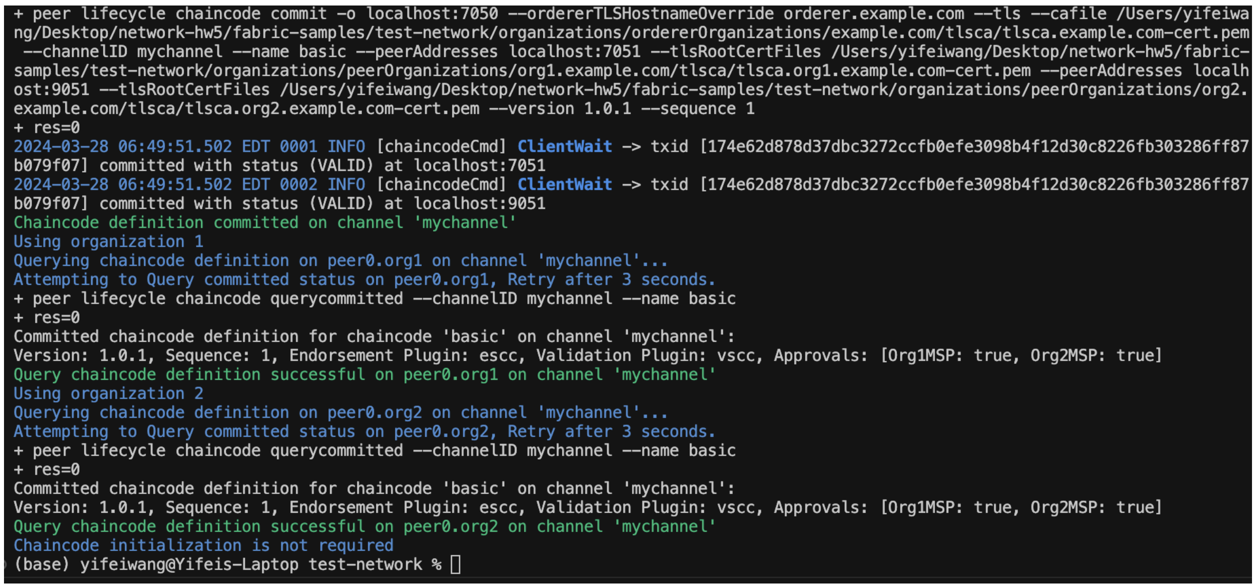
```
# Bring up the network and create a channel
./network.sh up createChannel -c mychannel -ca
```

Then this output deploys my chaincode to each peer nodes.



```
# Deploy the chaincode
./network.sh deployCC -ccn basic -ccp ../voting_app/chaincode-typescript/ -ccl typescript
```

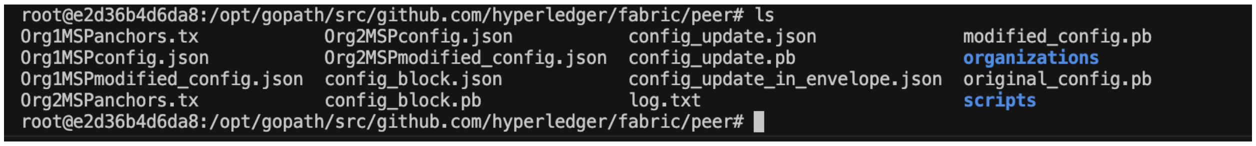
This is an example screenshot of the output



```
+ peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /Users/yifeiwa
ng/Desktop/network-hw5/fabric-samples/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem
 --channelID mychannel --name basic --peerAddresses localhost:7051 --tlsRootCertFiles /Users/yifeiwang/Desktop/network-hw5/fabric-
samples/test-network/organizations/peerOrganizations/org1.example.com/tlsca/tlsca.org1.example.com-cert.pem --peerAddresses localh
ost:9051 --tlsRootCertFiles /Users/yifeiwang/Desktop/network-hw5/fabric-samples/test-network/organizations/peerOrganizations/org2.
example.com/tlsca/tlsca.org2.example.com-cert.pem --version 1.0.1 --sequence 1
+ res=0
2024-03-28 06:49:51.502 EDT 0001 INFO [chaincodeCmd] ClientWait -> txid [174e62d878d37dbc3272ccfb0efe3098b4f12d30c8226fb303286ff87
b079f07] committed with status (VALID) at localhost:7051
2024-03-28 06:49:51.502 EDT 0002 INFO [chaincodeCmd] ClientWait -> txid [174e62d878d37dbc3272ccfb0efe3098b4f12d30c8226fb303286ff87
b079f07] committed with status (VALID) at localhost:9051
Chaincode definition committed on channel 'mychannel'
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to Query committed status on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0.1, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to Query committed status on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0.1, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Chaincode initialization is not required
(base) yifeiwang@Yifeis-Laptop test-network % 
```

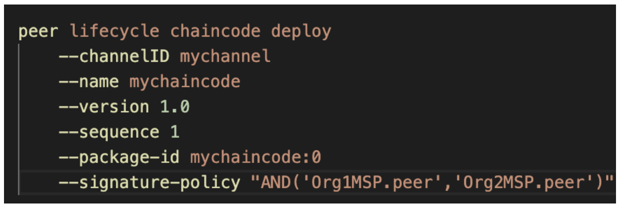
Note that the endorsement policy for the current basic network is that only peer0 from org1 must endorse before commit. If I want to make modifications to the endorsement policy, I would need to go into the CLI node and run the peer cli to deploy my chaincode and define my endorsement policy. This is the node that fabric is specifically made for the administrator to modify endorsement policy, update other peer node config, and manage channels.

I can go into the CLI node by using this command: docker exec -it cli bash



```
root@e2d36b4d6da8:/opt/gopath/src/github.com/hyperledger/fabric/peer# ls
Org1MSPanchors.tx            Org2MSPconfig.json           config_update.json            modified_config.pb
Org1MSPconfig.json           Org2MSPmodified_config.json  config_update.pb              organizations
Org1MSPmodified_config.json  config_block.json            config_update_in_envelope.json original_config.pb
Org2MSPanchors.tx            config_block.pb              log.txt                       scripts
root@e2d36b4d6da8:/opt/gopath/src/github.com/hyperledger/fabric/peer# 
```

This screenshot shows inside the CLI peer node. I would first need to generate a crypto material for each peer through the cryptogen cli.
Then, If I run the command below in the container, I would be able to define my signature policy.

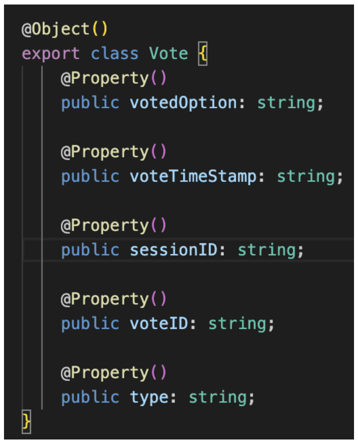


```
peer lifecycle chaincode deploy
    --channelID mychannel
    --name mychaincode
    --version 1.0
    --sequence 1
    --package-id mychaincode:0
    --signature-policy "AND('Org1MSP.peer','Org2MSP.peer')"
```

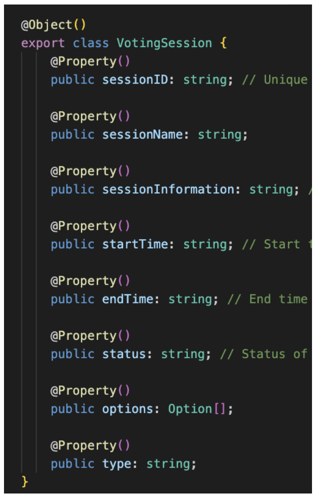
The –signature-policy flag defines the policy, in this case, "AND('Org1MSP.peer','Org2MSP.peer')" means that the two peers need to come to consent before endorsing. At this point, the chaincode deployment is finished.

2. Chaincode details

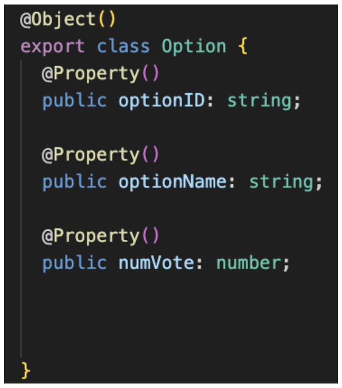
In my chaincode, I defined 3 types of assets: vote, session and option, where vote and session are stored in the ledger, and option is used as a property in session object.



```
@Object()
export class Vote {
    @Property()
    public votedOption: string;

    @Property()
    public voteTimeStamp: string;

    @Property()
    public sessionID: string;

    @Property()
    public voteID: string;

    @Property()
    public type: string;
}
```



```
@Object()
export class VotingSession {
    @Property()
    public sessionID: string; // Unique

    @Property()
    public sessionName: string;

    @Property()
    public sessionInformation: string; /

    @Property()
    public startTime: string; // Start t

    @Property()
    public endTime: string; // End time

    @Property()
    public status: string; // Status of

    @Property()
    public options: Option[];

    @Property()
    public type: string;
}
```

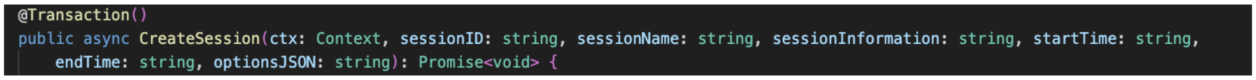


```
@Object()
export class Option {
    @Property()
    public optionID: string;

    @Property()
    public optionName: string;

    @Property()
    public numVote: number;


}
```

I also defined many types of transactions in my chaincode, such as read vote, read session, update vote, update sessions, create vote, create sessions, delete vote, delete sessions, and a get all asset function. These functions are deployed to all peer code and waiting to be run when called through fabric contract nodeJS API.

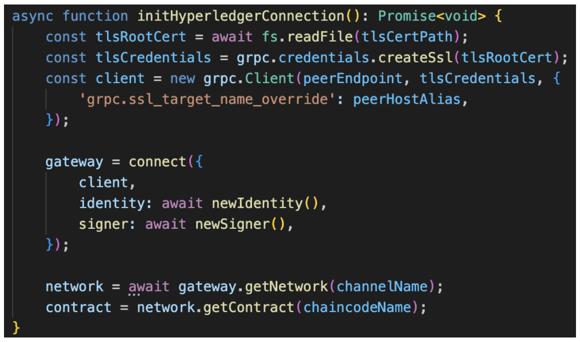
3. Fabric contract API- NodeJS and Backend

After developing and deploying chaincode, I would call them through the fabric contract API. This would be done by first setting up a connection to the hyperledger(called gateway), then using the gateway to get the network, then getting the contract API. Fabric contract API allows me to call my chaincode functions through .evaluateTransaction(). For example, if I want to call the CreateSession transaction defined in my chain code, as shown below:



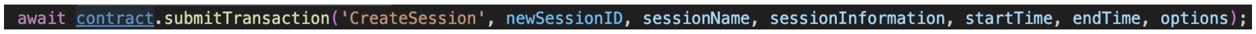
```
@Transaction()
public async CreateSession(ctx: Context, sessionID: string, sessionName: string, sessionInformation: string, startTime: string,
    endTime: string, optionsJSON: string): Promise<void> {
```

I would first setup connection:



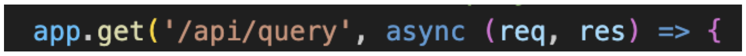
```
async function initHyperledgerConnection(): Promise<void> {
    const tlsRootCert = await fs.readFile(tlsCertPath);
    const tlsCredentials = grpc.credentials.createSsl(tlsRootCert);
    const client = new grpc.Client(peerEndpoint, tlsCredentials, {
        'grpc.ssl_target_name_override': peerHostAlias,
    });

    gateway = connect({
        client,
        identity: await newIdentity(),
        signer: await newSigner(),
    });

    network = await gateway.getNetwork(channelName);
    contract = network.getContract(chaincodeName);
}
```

Then call the function through:

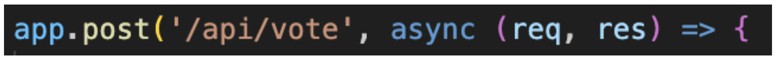


```
await contract.submitTransaction('CreateSession', newSessionID, sessionName, sessionInformation, startTime, endTime, options);
```

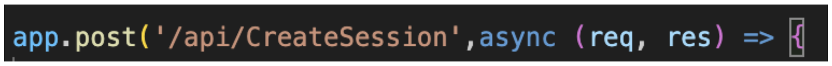
Using this contract API, I created a rest API for connecting my backend to front end. My restAPI contains 3 functions, one for querying sessions, one for voting, and one for creating sessions.



```
app.get('/api/query', async (req, res) => {
```



```
app.post('/api/vote', async (req, res) => {
```



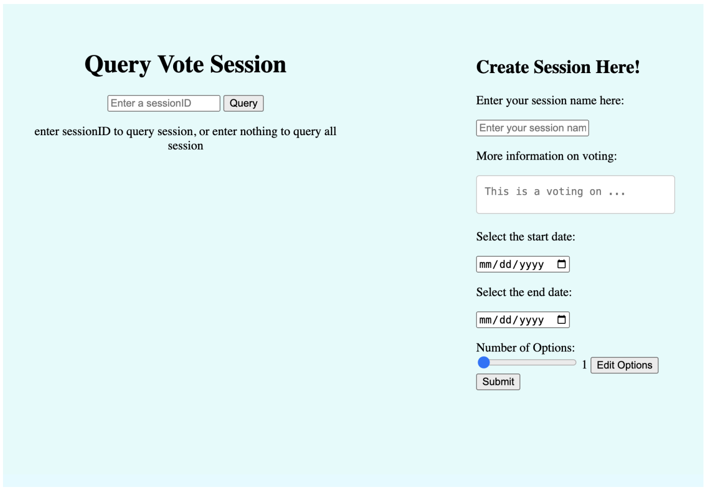
```
app.post('/api/CreateSession',async (req, res) => {
```

With these 3 APIs, I can submit voting, create voting sessions, and query my hyperledger through simple API calls from the frontend.

4. Express js for front end

I developed my frontend using express js. Some of the user interfaces are shown below:

This is the home page, it contains 2 section that allows user to either query voting sessions or create session:



Query Vote Session

Enter a sessionID [Query]

enter sessionID to query session, or enter nothing to query all session

Create Session Here!

Enter your session name here:

Enter your session nam

More information on voting:

This is a voting on ...

Select the start date:

mm/dd/yyyy

Select the end date:

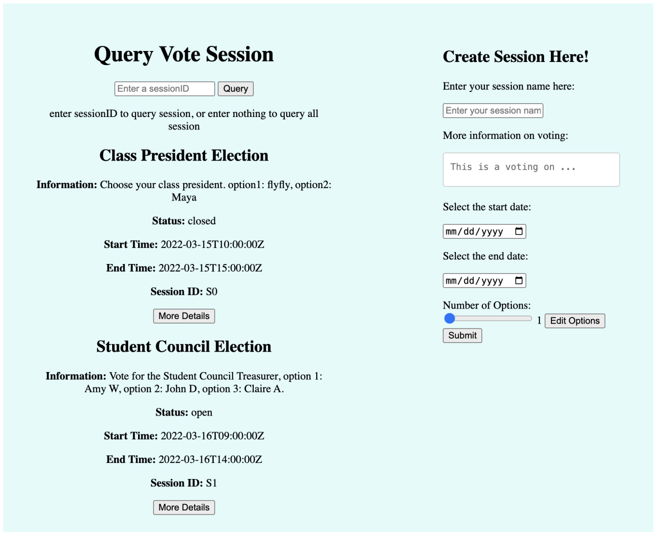
mm/dd/yyyy

Number of Options:

1 [Edit Options]

[Submit]

If the user click query without enter session ID, the page will return all sessions, open or closed:



Query Vote Session

Enter a sessionID [Query]

enter sessionID to query session, or enter nothing to query all session

Class President Election

Information: Choose your class president. option1: flyfly, option2: Maya

Status: closed

Start Time: 2022-03-15T10:00:00Z

End Time: 2022-03-15T15:00:00Z

Session ID: S0

[More Details]

Student Council Election

Information: Vote for the Student Council Treasurer, option 1: Amy W, option 2: John D, option 3: Claire A.

Status: open

Start Time: 2022-03-16T09:00:00Z

End Time: 2022-03-16T14:00:00Z

Session ID: S1

[More Details]

Create Session Here!

Enter your session name here:

Enter your session nam

More information on voting:

This is a voting on ...

Select the start date:

mm/dd/yyyy

Select the end date:

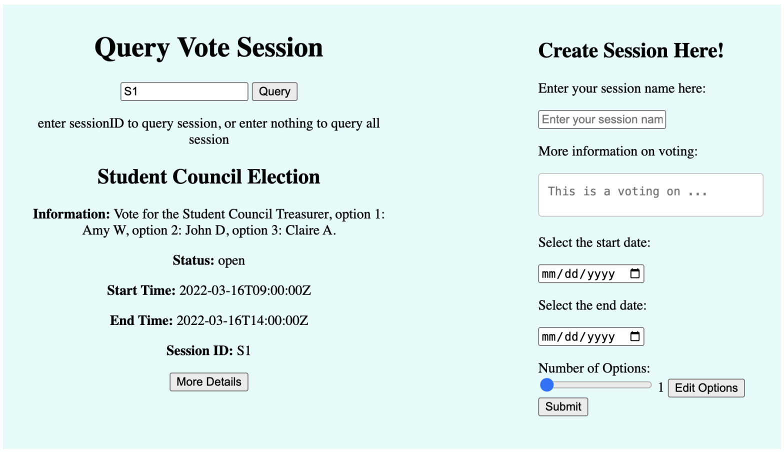
mm/dd/yyyy

Number of Options:

1 [Edit Options]

[Submit]

User can also query by entering session ID:


Query Vote Session

S1 Query

enter sessionID to query session, or enter nothing to query all session

Student Council Election

Information: Vote for the Student Council Treasurer, option 1: Amy W, option 2: John D, option 3: Claire A.

Status: open

Start Time: 2022-03-16T09:00:00Z

End Time: 2022-03-16T14:00:00Z

Session ID: S1

More Details

Create Session Here!

Enter your session name here:

Enter your session nam

More information on voting:

This is a voting on ...

Select the start date:

mm/dd/yyyy

Select the end date:

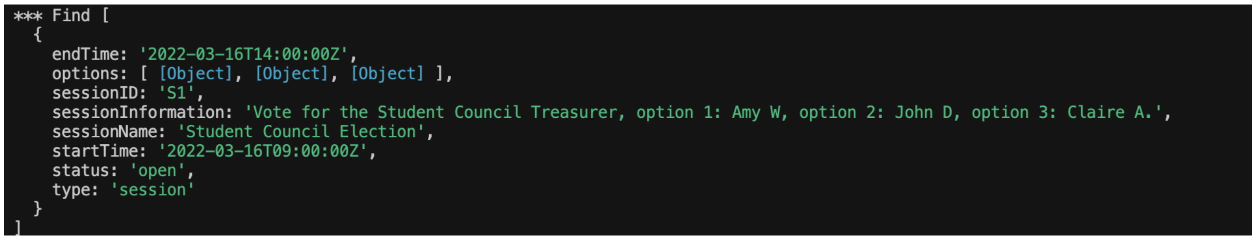
mm/dd/yyyy

Number of Options:

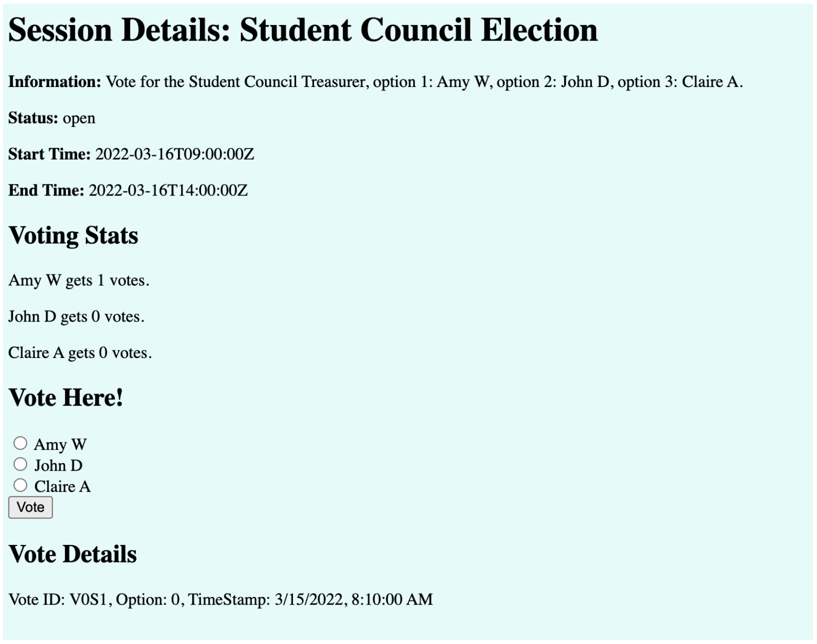
1 Edit Options

Submit

The console looks like this when querying session ‘S1’:


```
*** Find [
  {
    endTime: '2022-03-16T14:00:00Z',
    options: [ [Object], [Object], [Object] ],
    sessionID: 'S1',
    sessionInformation: 'Vote for the Student Council Treasurer, option 1: Amy W, option 2: John D, option 3: Claire A.',
    sessionName: 'Student Council Election',
    startTime: '2022-03-16T09:00:00Z',
    status: 'open',
    type: 'session'
  }
]
```

Then, if you click on ‘More Details’, there will be information on the voting session, and if the vote is currently open, you should be able to vote by selecting a choice and clicking vote. ‘Vote Details’ show all vote ID and timestamps.


Session Details: Student Council Election

Information: Vote for the Student Council Treasurer, option 1: Amy W, option 2: John D, option 3: Claire A.

Status: open

Start Time: 2022-03-16T09:00:00Z

End Time: 2022-03-16T14:00:00Z

Voting Stats

Amy W gets 1 votes.

John D gets 0 votes.

Claire A gets 0 votes.

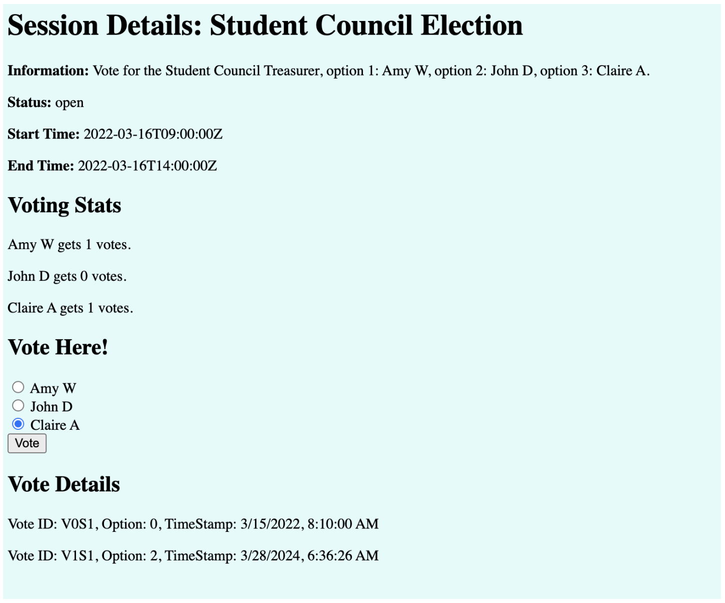
Vote Here!

○ Amy W
○ John D
○ Claire A
Vote

Vote Details

Vote ID: V0S1, Option: 0, TimeStamp: 3/15/2022, 8:10:00 AM

After clicking vote for ‘Claire A’, notice the vote num updates and vote details updates.



Session Details: Student Council Election

Information: Vote for the Student Council Treasurer, option 1: Amy W, option 2: John D, option 3: Claire A.

Status: open

Start Time: 2022-03-16T09:00:00Z

End Time: 2022-03-16T14:00:00Z

Voting Stats

Amy W gets 1 votes.

John D gets 0 votes.

Claire A gets 1 votes.

Vote Here!

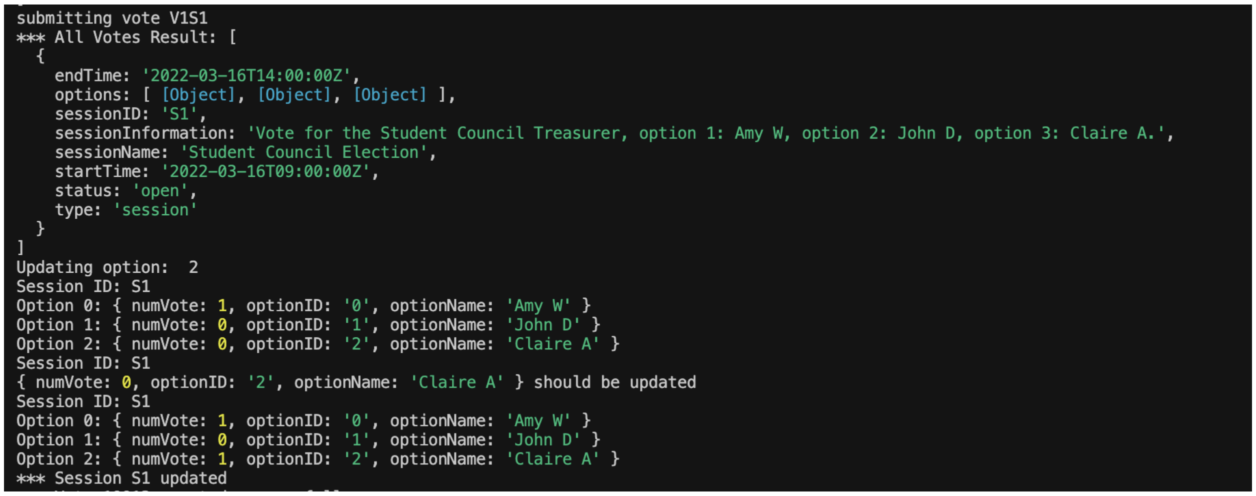
Amy W
John D
[✓ Claire A]
Vote

Vote Details

Vote ID: V0S1, Option: 0, TimeStamp: 3/15/2022, 8:10:00 AM

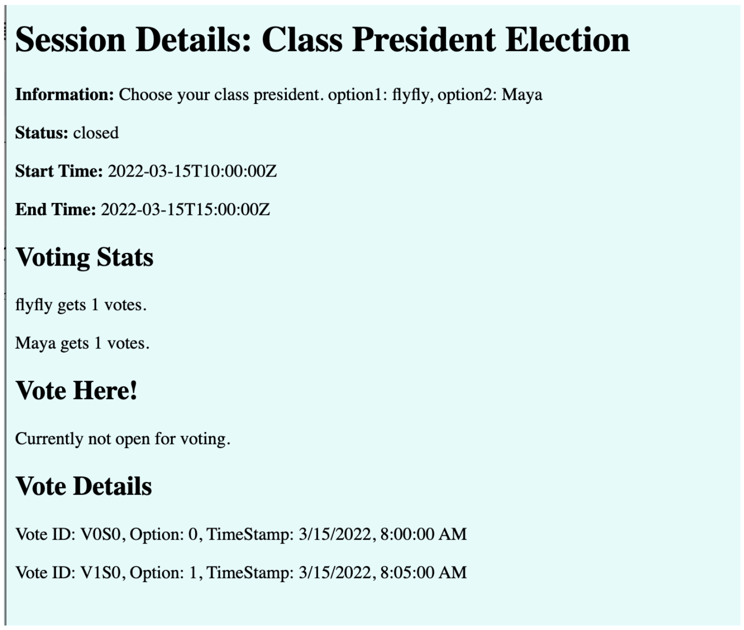
Vote ID: V1S1, Option: 2, TimeStamp: 3/28/2024, 6:36:26 AM

Console when voing for Claire A, shows the vote session being updated on, option being updated on, and message on



```
submitting vote V1S1
*** All Votes Result: [
  {
    endTime: '2022-03-16T14:00:00Z',
    options: [ [Object], [Object], [Object] ],
    sessionID: 'S1',
    sessionInformation: 'Vote for the Student Council Treasurer, option 1: Amy W, option 2: John D, option 3: Claire A.',
    sessionName: 'Student Council Election',
    startTime: '2022-03-16T09:00:00Z',
    status: 'open',
    type: 'session'
  }
]
Updating option:  2
Session ID: S1
Option 0: { numVote: 1, optionID: '0', optionName: 'Amy W' }
Option 1: { numVote: 0, optionID: '1', optionName: 'John D' }
Option 2: { numVote: 0, optionID: '2', optionName: 'Claire A' }
Session ID: S1
{ numVote: 0, optionID: '2', optionName: 'Claire A' } should be updated
Session ID: S1
Option 0: { numVote: 1, optionID: '0', optionName: 'Amy W' }
Option 1: { numVote: 0, optionID: '1', optionName: 'John D' }
Option 2: { numVote: 1, optionID: '2', optionName: 'Claire A' }
*** Session S1 updated
```

Sessions should be closed if the voting time passed or not yet started.


Session Details: Class President Election

Information: Choose your class president. option1: flyfly, option2: Maya

Status: closed

Start Time: 2022-03-15T10:00:00Z

End Time: 2022-03-15T15:00:00Z

Voting Stats

flyfly gets 1 votes.

Maya gets 1 votes.

Vote Here!

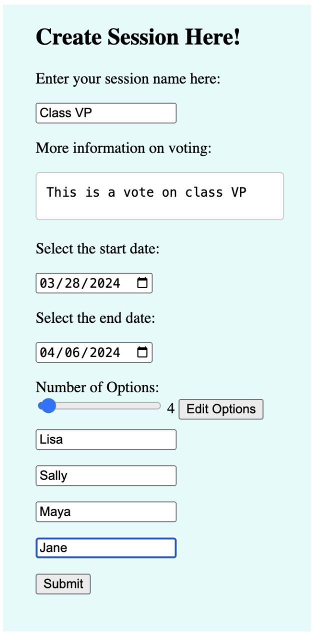
Currently not open for voting.

Vote Details

Vote ID: V0S0, Option: 0, TimeStamp: 3/15/2022, 8:00:00 AM

Vote ID: V1S0, Option: 1, TimeStamp: 3/15/2022, 8:05:00 AM

User can also submit a form to create a voting session:


Create Session Here!

Enter your session name here:

Class VP

More information on voting:

This is a vote on class VP

Select the start date:

03/28/2024

Select the end date:

04/06/2024

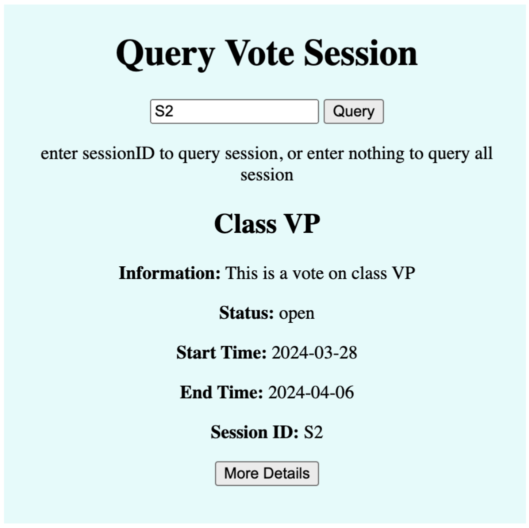
Number of Options: 4

Lisa

Sally

Maya

Jane


Query Vote Session

S2

enter sessionID to query session, or enter nothing to query all session

Class VP

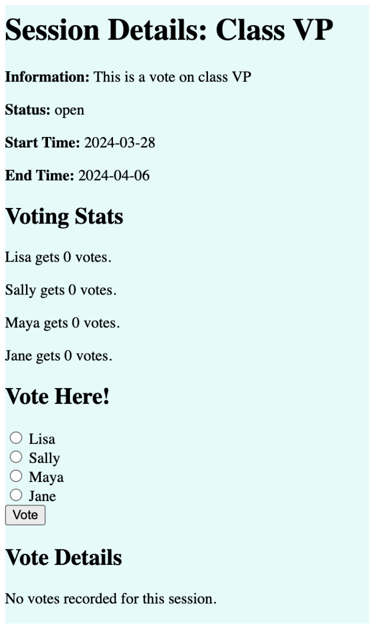
Information: This is a vote on class VP

Status: open

Start Time: 2024-03-28

End Time: 2024-04-06

Session ID: S2


Session Details: Class VP

Information: This is a vote on class VP

Status: open

Start Time: 2024-03-28

End Time: 2024-04-06

Voting Stats

Lisa gets 0 votes.

Sally gets 0 votes.

Maya gets 0 votes.

Jane gets 0 votes.

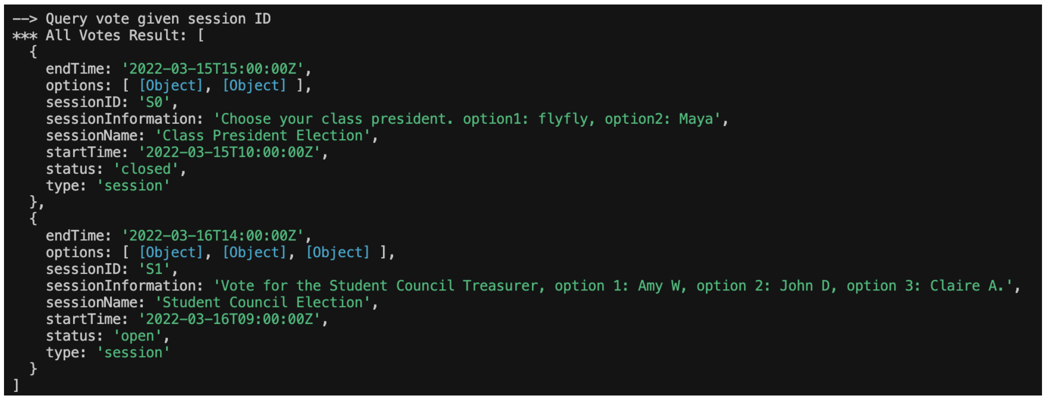
Vote Here!

○ Lisa
○ Sally
○ Maya
○ Jane

Vote Details

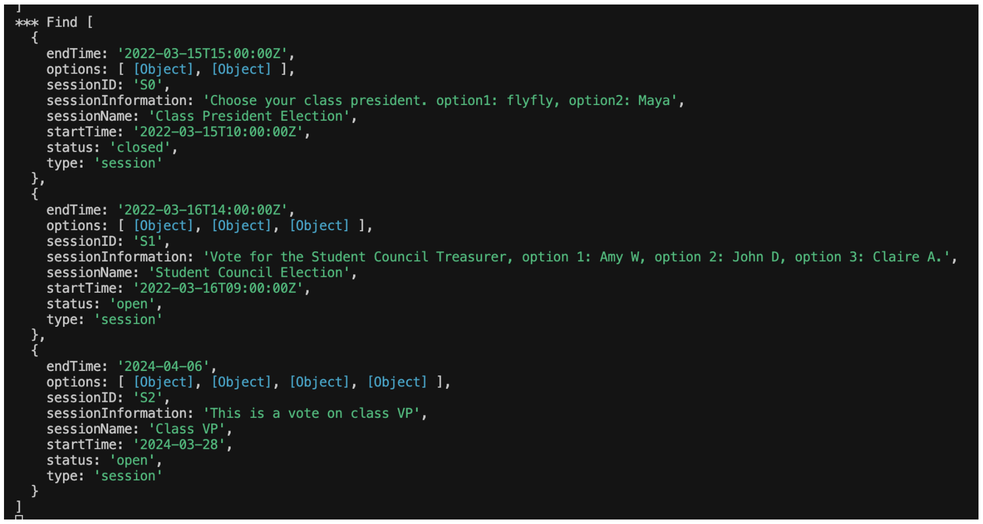
No votes recorded for this session.

This is the output from query all session before session created:



```
--> Query vote given session ID
*** All Votes Result: [
  {
    endTime: '2022-03-15T15:00:00Z',
    options: [ [Object], [Object] ],
    sessionID: 'S0',
    sessionInformation: 'Choose your class president. option1: flyfly, option2: Maya',
    sessionName: 'Class President Election',
    startTime: '2022-03-15T10:00:00Z',
    status: 'closed',
    type: 'session'
  },
  {
    endTime: '2022-03-16T14:00:00Z',
    options: [ [Object], [Object], [Object] ],
    sessionID: 'S1',
    sessionInformation: 'Vote for the Student Council Treasurer, option 1: Amy W, option 2: John D, option 3: Claire A.',
    sessionName: 'Student Council Election',
    startTime: '2022-03-16T09:00:00Z',
    status: 'open',
    type: 'session'
  }
]
```

This is the output after, you can see the third session is newly created.



```
*** Find [
  {
    endTime: '2022-03-15T15:00:00Z',
    options: [ [Object], [Object] ],
    sessionID: 'S0',
    sessionInformation: 'Choose your class president. option1: flyfly, option2: Maya',
    sessionName: 'Class President Election',
    startTime: '2022-03-15T10:00:00Z',
    status: 'closed',
    type: 'session'
  },
  {
    endTime: '2022-03-16T14:00:00Z',
    options: [ [Object], [Object], [Object] ],
    sessionID: 'S1',
    sessionInformation: 'Vote for the Student Council Treasurer, option 1: Amy W, option 2: John D, option 3: Claire A.',
    sessionName: 'Student Council Election',
    startTime: '2022-03-16T09:00:00Z',
    status: 'open',
    type: 'session'
  },
  {
    endTime: '2024-04-06',
    options: [ [Object], [Object], [Object], [Object] ],
    sessionID: 'S2',
    sessionInformation: 'This is a vote on class VP',
    sessionName: 'Class VP',
    startTime: '2024-03-28',
    status: 'open',
    type: 'session'
  }
]
```

Analysis on result lessons learned:

This is a very great learning experience for me. During the learning phase, I spent a long time reading documents, watching youtube videos and reading code to learn the basic structure and workflow of fabric. It is very complicated and involves many new concepts, but I managed to get through all of them. In the beginning of development, I was very confused, learning through modifying the sample code for the contract API and running. This is very helpful as it helps me to understand how to interact with the contract API. Then, after understanding the contract API, I trace back to the chaincode functions and start my vote system development. When working with chaincode, debugging was a pain, because I did not figure out how to log on the console(later I knows that it would output to the peer node container, which I could check by docker log), but the debugging output is still very limited and does not gives much information. Therefore I implemented a try catch block to catch potential errors, which helps me a lot when debugging.

I believe I've implemented the core functionalities of a voting app, including voting, creating voting sessions, and querying sessions. I believe the rest API and user interface I developed is also clear. This is a great learning opportunity, because I have not worked with full stack development a lot in the past, though this is a small web app, I spent a lot of time getting familiar with express js, learning the grammar and debugging the app. Overall, this is a very meaningful experience for me, and boosted my understanding on both hyperledger and app development.