Solidity Assignment Phase-3

(Deploying Smart contract to geth using truffle)

Submitted by : Manikandan Ramesh

**Initialize private blockchain using geth:**

**Create folder for datadir**

$ mkdir datadir1

Initalize private blockchain

$ geth --datadir datadir1/ init genesis.json

**Start geth console and unlock accounts for transactions**

Create 3 accounts after successfully launching a geth client

$ geth --datadir datadir1/ --networkid 2019 –rpc –rpcport 8545 –-allow-insecure-unlock console

Creating Account using command

> personal.newAccount('test') **(**need to create 3 account)

Unlock Account using comman

> personal.unlockAccount ("0xd30dbca7cad74ff17be201f30167e7437eb10c6f”,'test',0) **(**need to unlock 3 account)

> personal.unlockAccount ("0x6d6aaf9e12c3caab8900748f7a36fbe8276a087f",'test',0) **(**need to unlock 3 account)

> personal.unlockAccount ("0xf245b64625d15660b3243f789bdf139210ea304b",'test',0) **(**need to unlock 3 account)

Start miner in geth console

miner.start()

Truffle commands for KYC deployment and interacting with contract

**Compile smart contarct**

$ truffle compile

**Migrate contract to our geth private network**

$ truffle migrate

Truffle console interaction with KYC contract

**Truffle Console start**

$ truffle console

**Create instance of kyc contract.**

truffle(development)> let kyc = await KYCContract.at("0xBf7C0bA17BF612a3D95515019E22a21589Df6C8c")

**Saving all accounts from private blockchain to accounts variable to further use**

truffle(development)> var accounts;

truffle(development)> web3.eth.getAccounts(function(err,res) { accounts = res; });

ADMIN INTERACTION

**Create Bank1**

truffle(development)> kyc.addBank("sbi","sbi1",accounts[1],true)

**Create Bank2**

truffle(development)> kyc.addBank("axis","axis1",accounts[2],true)

**Get the Bank name using bank.ethaddress**

truffle(development)> kyc.getBankName(accounts[1])

truffle(development)> kyc.getBankName(accounts[2])

**See the bank created using above command**

truffle(development)> kyc.allBanks(0)

BANK INTERACTIONS:

|  |
| --- |
| ***Note:***  *Hence using from:accounts[1] which contains banks address* |

**Add Customer**

truffle(development)> kyc.addCustomer("mani","32ead53416df3093100403ee44d5eb045ed5bb3b5105dafa3e137a359f60cbc7",{from:accounts[1]})

**To check if customer is added**

truffle(development)> kyc.allCustomers(0)

**Updating Customer information**

truffle(development)> kyc.modifyCustomer("mani","f61622fdc7ab3ac33b32335b853f9889d486a0c5b2493353470ae3874c8bda0f",{from:accounts[1]})

**To Check data is updated**

truffle(development)> kyc.allCustomers(0)

**Adding KYC Request**

truffle(development)> kyc.addRequest("sumeet","0xd30dbca7cad74ff17be201f30167e7437eb10c6f",{from:accounts[1]})

**To See if request is added**

truffle(development)> kyc.allRequests(0)

**Update KYC Votes**

truffle(development)> kyc.updateKYCVotes("mani",true,{from:accounts[1]})

**To See upvotes count**

truffle(development)> kyc.allCustomers(0)

**To See if record is moved to validKYCs**

truffle(development)> kyc.validKYCs(0)

**Updating KYC votes with false (down voting)**

truffle(development)> kyc.updateKYCVotes("mani",false,{from:accounts[1]})

**To See upvotes count**

truffle(development)> kyc.allCustomers(0)

**Other banks adding rating to bank**

truffle(development)> kyc.addRatingToBank(accounts[1])

**View Document hash of customer**

truffle(development)> kyc.viewCustomer("mani")

**Set Password to customer data**

truffle(development)> kyc.setCustomerPassword("mani","password",{from:accounts[1]})

**See the password set.**

truffle(development)> kyc.allCustomers(0)

**View Customer Rating provided by banks**

truffle(development)> await kyc.getCustomersRating("mani",{from:accounts[2]})

**View Bank Ratings**

truffle(development)> kyc.getBanksRating(accounts[1],{from:accounts[1]})

**Add Customer Ratings**

truffle(development)> kyc.addRatingToCustomer("mani",{from:accounts[1]})

**View Customer Ratings**

truffle(development)> kyc.getCustomersRating("mani",{from:accounts[1]})

**Remove KYC Request**

truffle(development)> kyc.removeRequest("mani",{from:accounts[1]})

**Remove Customer**

truffle(development)> kyc.removeCustomer("mani",{from:accounts[1]})

**Remove Bank by admin(contract owner)**

truffle(development)> kyc.removeBank(accounts[1])