**Initialize private blockchain using geth:**

**Create folder for datadir**

$ mkdir datadir3

Initalize private blockchain

$ geth --datadir datadir3/ init genesis.json

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

Create 3 accounts after successfully launching a geth client

$ geth --datadir datadir1/ --networkid 2020 –rpc –rpcport 8545 –-allow-insecure-unlock console

Creating Account using command

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

Unlock Account using comman

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

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

> personal.unlockAccount ("0xf245b64625d15660b3243f789bdf139210ea304b",'testAccount',0) **(**need to unlock all 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.**

“ Address to be used is smart contract migrated address”

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",accounts[1],"sbi1")

**Create Bank2**

truffle(development)> KYC.addBank("axis",accounts[2],”axis1”)

**See the bank created using above command**

truffle(development)> KYC.allbanks(0)

BANK INTERACTIONS:

**Add Customer**

truffle(development)> KYC.addCustomer("sharansh","32ead53416df3093100403ee44d5eb045ed5bb3b5105dafa3e137a359f60cbc7")

**To check if customer is added**

truffle(development)> KYC.allCustomers(0)

**Updating Customer information**

truffle(development)> KYC.modifyCustomer("sharansh","f61622fdc7ab3ac33b32335b853f9889d486a0c5b2493353470ae3874c8bda0f")

**To Check data is updated**

truffle(development)> KYC.allCustomers(0)

**Adding KYC Request**

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

**To See if request is added**

truffle(development)> KYC.allRequests(0)

**Upvote Bank**

truffle(development)> KYC.UpvoteBank({from:accounts[1]})

**To See upvotes count**

truffle(development)> KYC.allCustomers(0)

**View Document hash of customer**

truffle(development)> KYC.viewCustomer("sharansh")

**Set Password to customer data**

truffle(development)> KYC.setCustomerPassword("sharansh","password")

**See the password set.**

truffle(development)> KYC.allCustomers(0)

**View Customer Rating provided by banks**

truffle(development)> await KYC.getCustomerRating("sharansh")

**View Bank Ratings**

truffle(development)> KYC.getBankRating(accounts[1])

**Add Customer Ratings**

truffle(development)> KYC.UpvoteCustomer("sharansh")

**View Customer Ratings**

truffle(development)> KYC.getCustomerRating("sharansh")

**Remove KYC Request**

truffle(development)> KYC.removeRequest("sharan")

**Remove Customer**

truffle(development)> KYC.removeCustomer("sharansh")

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

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