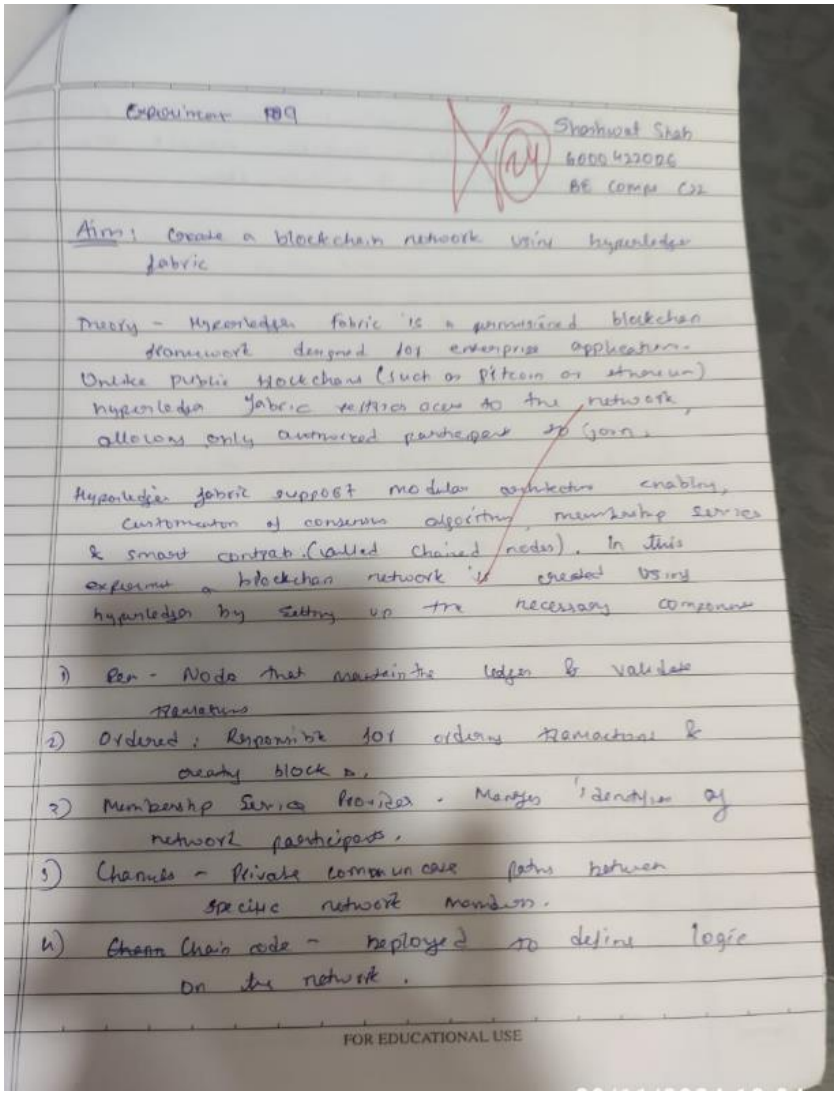
BLOCKCHAIN TECHNOLOGY

Name:-Preksha Ashok Patel
Sapid:-60004210126
Branch:-Computer Engineering
Div/Batch:-C2-1

EXPERIMENT NO.09


Experiment 09

Shashwat Shah
60004220006
BE Comps C2

Aim: Create a blockchain network using hyperledger fabric

Theory - Hyperledger fabric is a permissioned blockchain framework designed for enterprise applications. Unlike public blockchains (such as Bitcoin or Ethereum) hyperledger fabric restricts access to the network, allowing only authorized participants to join.

Hyperledger fabric support modular architecture enabling, customization of consensus algorithm, membership services & smart contracts (called chained nodes). In this experiment a blockchain network is created using hyperledger by setting up the necessary components

1) Peer - Node that maintain the ledger & validate transactions
2) Orderer : Responsible for ordering transactions & creating blocks.
3) Membership Service Provider - Manages identities of network participants.
4) Channels - Private communication paths between specific network members.
5) Chain code - Deployed to define logic on the network.

FOR EDUCATIONAL USE

CODE & OUTPUT :-

Chaincode file:-

```
'use strict';

const { Contract } = require('fabric-contract-api');

class AssetTransfer extends Contract {
```

```
    async InitLedger(ctx) {
        const assets = [
            { ID: 'asset1', Color: 'blue', Size: 5, Owner: 'Tomoko', AppraisedValue: 300 },
            { ID: 'asset2', Color: 'red', Size: 3, Owner: 'Brad', AppraisedValue: 400 },
        ];

        for (const asset of assets) {
            await ctx.stub.putState(asset.ID, Buffer.from(JSON.stringify(asset)));
        }
        return 'Ledger initialized!';
    }

    async CreateAsset(ctx, id, color, size, owner, appraisedValue) {
        const asset = { ID: id, Color: color, Size: size, Owner: owner, AppraisedValue:
parseInt(appraisedValue) };
        await ctx.stub.putState(id, Buffer.from(JSON.stringify(asset)));
        return `Asset ${id} created!`;
    }

    async ReadAsset(ctx, id) {
        const assetJSON = await ctx.stub.getState(id);
        if (!assetJSON || assetJSON.length === 0) {
            throw new Error(`The asset ${id} does not exist`);
        }
        return assetJSON.toString();
    }

    async TransferAsset(ctx, id, newOwner) {
        const assetString = await this.ReadAsset(ctx, id);
        const asset = JSON.parse(assetString);
        asset.Owner = newOwner;

        await ctx.stub.putState(id, Buffer.from(JSON.stringify(asset)));
        return `Asset ${id} ownership transferred to ${newOwner}!`;
    }
}

module.exports = AssetTransfer;
```

App.js file code:-

```
const { Gateway, Wallets } = require('fabric-network');
const path = require('path');
const fs = require('fs');

async function main() {
    try {
        // Load network configuration
```

```
        const ccpPath = path.resolve(__dirname, '..', 'fabric-samples', 'test-network',
'organizations', 'peerOrganizations', 'org1.example.com', 'connection-org1.json');
        const ccp = JSON.parse(fs.readFileSync(ccpPath, 'utf8'));

        // Create wallet
        const walletPath = path.join(process.cwd(), 'wallet');
        const wallet = await Wallets.newFileSystemWallet(walletPath);

        // Check for user identity
        const userExists = await wallet.get('appUser');
        if (!userExists) {
            console.log('User identity not found! Register and enroll the user first.');
            return;
        }

        // Connect to the gateway
        const gateway = new Gateway();
        await gateway.connect(ccp, { wallet, identity: 'appUser', discovery: { enabled: true,
asLocalhost: true } });

        // Get the network and contract
        const network = await gateway.getNetwork('mychannel');
        const contract = network.getContract('basic');

        // Example interactions
        console.log('Initializing ledger...');
        await contract.submitTransaction('InitLedger');

        console.log('Creating a new asset...');
        await contract.submitTransaction('CreateAsset', 'asset3', 'green', '10', 'Preksha', '500');

        console.log('Reading asset...');
        const result = await contract.evaluateTransaction('ReadAsset', 'asset3');
        console.log(`Asset details: ${result.toString()}`);

        console.log('Transferring asset ownership...');
        await contract.submitTransaction('TransferAsset', 'asset3', 'NewOwner');

        console.log('Asset ownership updated.');
    } catch (error) {
        console.error(`Error: ${error}`);
    }
}

main();
```

OUTPUT:-

```
Initializing ledger...
Ledger initialized!

Creating a new asset...
Asset asset3 created!

Reading asset...
Asset details: {"ID":"asset3","Color":"green","Size":10,"Owner":"Preksha","AppraisedValue"

Transferring asset ownership...
Asset asset3 ownership transferred to NewOwner!

Asset ownership updated.
```