**Project Code**

const SHA256 = require( 'crypto-js/sha256' );

class Transaction {

constructor( timestamp, payerAddr, payeeAddr, amount ) {

this.timestamp = timestamp;

this.payerAddr = payerAddr;

this.payeeAddr = payeeAddr;

this.amount = amount;

}

}

class Block {

constructor( timestamp, txns, previousHash ) {

this.timestamp = timestamp;

this.txns = txns;

this.previousHash = previousHash;

this.nonce = 0;

this.hash = this.calculateHash();

}

calculateHash() {

return SHA256( this.index + this.previousHash + this.timestamp + JSON.stringify( this.data ) + this.nonce ).toString();

}

mineBlock( difficulty ) {

let count = 0;

while( this.hash.substring( 0, difficulty ) !== Array( difficulty + 1 ).join( "0" ) ) {

this.nonce++;

count++;

this.hash = this.calculateHash();

}

console.log( "Block successfully hashed (" + count + " iterations). Hash: " + this.hash );

}

}

class Blockchain {

constructor() {

this.chain = [];

this.difficulty = 3;

this.unminedTxns = [];

this.miningReward = 50;

this.registeredAddresses = [ 'wallet-Alice', 'wallet-Bob', 'wallet-Charlie', 'wallet-Miner49r' ];

this.createGenesisBlock();

this.airdropCoins( 100 );

}

airdropCoins( coins ) {

for ( const addr of this.registeredAddresses ) {

let txn = new Transaction( Date.now(), "mint", addr, coins );

this.unminedTxns.push( txn );

}

this.mineCurrentBlock( 'wallet-Miner49r' );

}

createGenesisBlock() {

let txn = new Transaction( Date.now(), "mint", "genesis", 0 );

let block = new Block( Date.now(), [ txn ], "0" );

this.chain.push( block );

}

getLatestBlock() {

return this.chain[ this.chain.length - 1 ];

}

mineCurrentBlock( minerAddr ) {

let validatedTxns = [];

for ( const txn of this.unminedTxns ) {

if ( txn.payerAddr === "mint" || this.validateTransaction( txn ) ) {

validatedTxns.push( txn );

}

}

console.log( "transactions validated: " + validatedTxns.length );

let block = new Block( Date.now(), validatedTxns, this.getLatestBlock().hash );

block.mineBlock( this.difficulty );

console.log( "Current Block successfully mined..." );

this.chain.push( block );

this.unminedTxns = [

new Transaction( Date.now(), "mint", minerAddr, this.miningReward )

];

}

validateTransaction( txn ) {

let payerAddr = txn.payerAddr;

let balance = this.getAddressBalance( payerAddr );

if ( balance >= txn.amount ) {

return true;

} else {

return false;

}

}

createTransaction( txn ) {

this.unminedTxns.push( txn );

}

getAddressBalance( addr ) {

let balance = 0;

for ( const block of this.chain ) {

for ( const txn of block.txns ) {

if ( txn.payerAddr === addr ) {

balance -= txn.amount;

}

if ( txn.payeeAddr === addr ) {

balance += txn.amount;

}

}

}

return balance;

}

isChainValid() {

for ( let i = 1; i < this.chain.length; i++ ) {

const currentBlock = this.chain[ i ];

const previousBlock = this.chain[ i - 1 ];

// validate data integrity

if ( currentBlock.hash !== currentBlock.calculateHash() ) {

return false;

}

// validate hash chain link

if ( currentBlock.previousHash !== previousBlock.hash ) {

return false

}

}

// all good, no manipulated data or bad links

return true;

}

}

let demoCoin = new Blockchain();

// 1st Block

demoCoin.createTransaction( new Transaction( Date.now(), 'wallet-Alice', 'wallet-Bob', 1000 ) );

demoCoin.createTransaction( new Transaction( Date.now(), 'wallet-Bob', 'wallet-Alice', 25 ) );

console.log( "\nMining a block" );

demoCoin.mineCurrentBlock( 'wallet-Miner49r' );

console.log( "\nBalance: Alice: ", + demoCoin.getAddressBalance( 'wallet-Alice' ) );

console.log( "\nBalance: Bob: ", + demoCoin.getAddressBalance( 'wallet-Bob' ) );

console.log( "\nBalance: Miner49r: ", + demoCoin.getAddressBalance( 'wallet-Miner49r' ) );

// 2nd Block

demoCoin.createTransaction( new Transaction( Date.now(), 'wallet-Alice', 'wallet-Bob', 50 ) );

demoCoin.createTransaction( new Transaction( Date.now(), 'wallet-Bob', 'wallet-Alice', 25 ) );

console.log( "\nMining a block" );

demoCoin.mineCurrentBlock( 'wallet-Miner49r' );

console.log( "\nBalance: Alice: ", + demoCoin.getAddressBalance( 'wallet-Alice' ) );

console.log( "\nBalance: Bob: ", + demoCoin.getAddressBalance( 'wallet-Bob' ) );

console.log( "\nBalance: Miner49r: ", + demoCoin.getAddressBalance( 'wallet-Miner49r' ) );