PRACTICAL 3

[CS601] – Cryptography and Blockchain

Date – 30/01/2023 | *By* Aishwarya Suryakant Waghmare, PRN – 2001106059

Title/Aim of the practical :

To create a blockchain with 3 blocks and check the validity of blockchain using SHA256.

Apparatus/Tools/ Resources used :

- Lecture Notes
- E-Resources
- E-Book
- Laptop
- Metamask
- Hyperledger

Theory of the practical :

- ✓ Blockchain is a time-stamped decentralized series of fixed records that contains data of any size is controlled by a large network of computers that are scattered around the globe and not owned by a single organization.
- ✓ Every block is secured and connected with each other using hashing technology which protects it from being tampered by an unauthorized person.
- ✓ Creating Blockchain using Python, mining new blocks, and displaying the whole blockchain:
  - ➢ The data will be stored in JSON format which is very easy to implement and easy to read. The data is stored in a block and the block contains multiple data.
  - ➢ Each and every minute multiple blocks are added and to differentiate one from the other we will use fingerprinting.
  - ➢ The fingerprinting is done by using hash and to be particular we will use the SHA256 hashing algorithm.
  - ➢ Every block will contain its own hash and also the hash of the previous function so that it cannot get tampered with.
  - ➢ This fingerprinting will be used to chain the blocks together. Every block will be attached to the previous block having its hash and to the next block by giving its hash.
  - ➢ The mining of the new block is done by giving successfully finding the answer to the proof of work.
  - ➢ To make mining hard the proof of work must be hard enough to get exploited.
  - ➢ After mining the block successfully, the block will then be added to the chain.
  - ➢ After mining several blocks the validity of the chain must be checked in order to prevent any kind of tampering with the blockchain.
  - ➢ Then the web app will be made by using Flask and deployed locally or publicly as per the need of the user.

Procedure of the Practical/ Codes:

package.json

```
{
  "name": "practical-3-crypto-chain",
  "version": "1.0.0",
  "description": "This is a project for my practical 3.",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Aishwarya Waghmare",
  "license": "MIT",
  "devDependencies": {
    "jest": "^29.4.1",
    "mocha": "^10.2.0"
  }
}
```

block.js

```
class Block {
    constructor({ timeStamp, lastHash, hash, data }) {
        this.timeStamp = timeStamp;
        this.lastHash = lastHash;
        this.hash = hash;
        this.data = data;
    }
}

const block1 = new Block({ timeStamp: '01/01/01', lastHash: "foo-lasthash", hash: "foo-hash", data: "foo-data" });
const block2 = new Block({ timeStamp: '01/01/99', lastHash: block1.lastHash, hash: "hoo-fash", data: "foo-data-2" });
const block3 = new Block({ timeStamp: '01/01/98', lastHash: block2.lastHash, hash: "ho-fas", data: "foo-data-3" });

console.log("Block 1:", block1, block2, block3);

console.log(require("./config"))
```

config.js

```
const GENSIS_DATA = {
   timeStamp: 1,
   lastHash: "-----",
   hash: "hash-one",
   data: []
}

module.exports = GENSIS_DATA;
```

block.test.js

```
// const hexToBinary = require("");
// const GENSIS_DATA = require("./config");

// describe("Block", () => {
//    const timeStamp = 2000;
//    const lastHash = 'foo-hash';
//    const hash = "bar-hash";
//    const data = ['blockchain', 'data'];
//    const nonce = 1;
//    const difficulty = 1;
//    const block = new Block({ timeStamp, lastHash, hash, data, nonce, difficulty });

//    it("has a timeStamp, lastHash, hash, and a data property", () => {
//       expect(block.timeStamp).toEqual(timeStamp);
//       expect(block.lastHash).toEqual(lastHash);
//       expect(block.hash).toEqual(hash);
//       expect(block.data).toEqual(data);
//       expect(block.nonce).toEqual(nonce);
//       expect(block.nonce).toEqual(nonce);

//    });

//    describe("genesis()", () => {
//       const gensisBlock = Block.gensis();
```

```
//        it("returns a Block instance", () => {
//            expect(gensisBlock instanceof Block).toBe(true);
//        });

//        it("returns the genesis data", () => {
//            expect(genesisBlock).toEqual(GENSIS_DATA);
//        });
//    });

//    describe()
// })

class Block {
    constructor(data, previousHash) {
        this.data = data;
        this.previousHash = previousHash;
        this.timestamp = new Date().getTime();
        this.hash = this.calculateHash();
    }

    calculateHash() {
        return require('crypto').createHash('sha256').update(this.data + this.previousHash +
this.timestamp).digest('hex')
    }
}

class Blockchain {
    constructor() {
        this.chain = [this.createGensisBlock()];
    }

    createGensisBlock() {
        return new Block('Gensis Block', '0');
    }

    getLatestBlock() {
        return this.chain[this.chain.length - 1];
    }
```

```
    addBlock(data) {
        const previousBlock = this.getLatestBlock();
        const block = new Block(data, previousBlock.hash);
        this.chain.push(block);
    }

    isChainValid() {
        for (let i = 1; i < this.chain.length; i++) {
            const currentBlock = this.chain[i];
            const previousBlock = this.chain[i - 1];

            if (currentBlock.hash !== currentBlock.calculateHash()) {
                return false;
            }

            if (currentBlock.previousHash !== previousBlock.hash) {
                return false;
            }
        }

        return true;
    }
}

const blockchain = new Blockchain();
blockchain.addBlock('First Block');
blockchain.addBlock('Second Block');
blockchain.addBlock('Third Block');

console.log(JSON.stringify(blockchain, null, 2));
console.log(`Is blockchain valid? ${blockchain.isChainValid()}`)
```

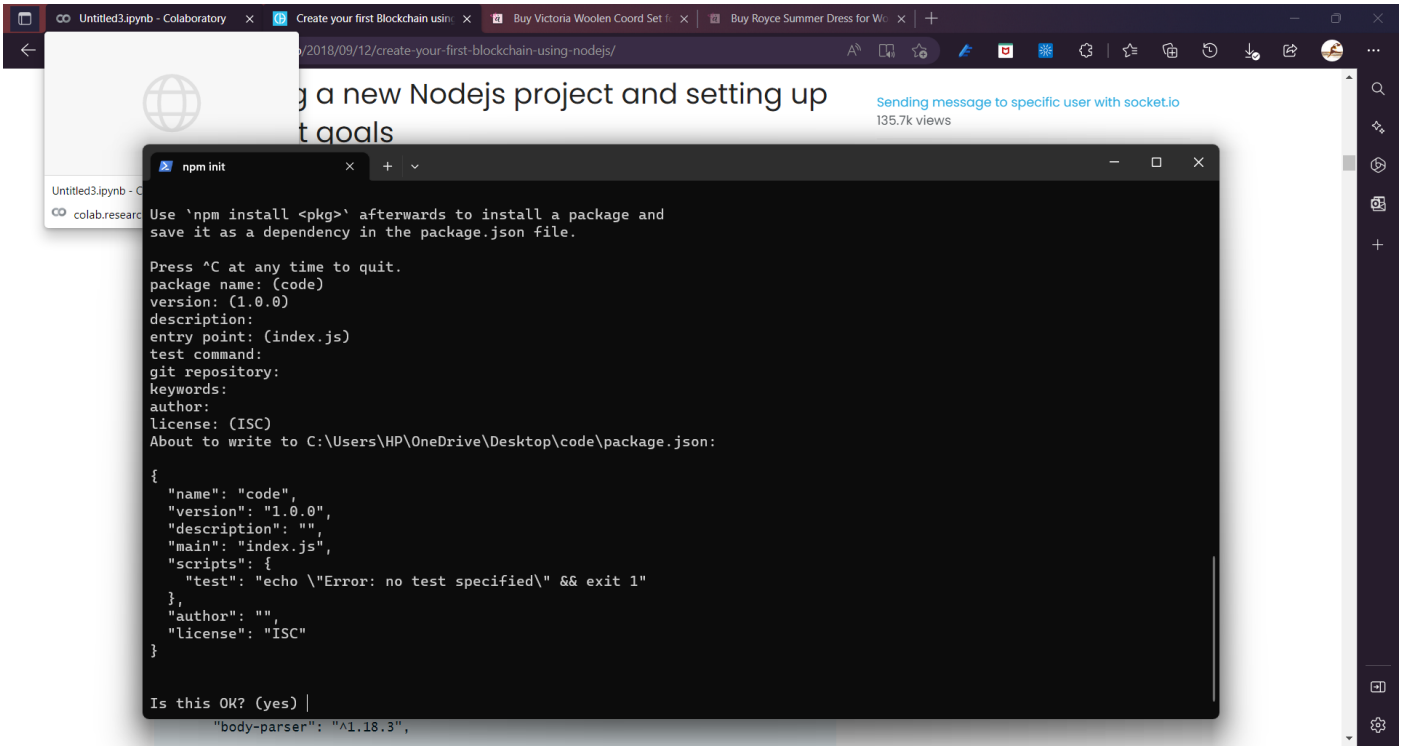
block_updated.js

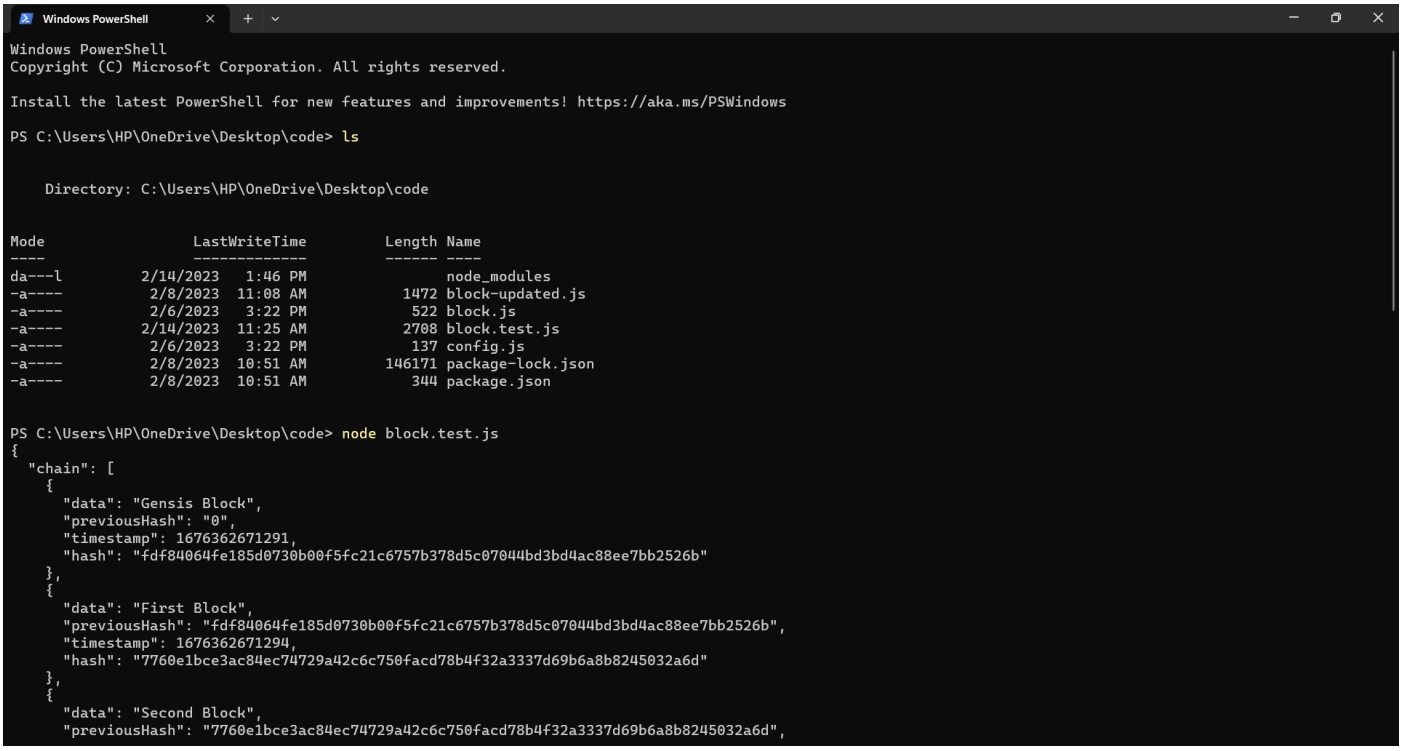
```
const hexToBinary = require("hex-to-binary")
const { GENSIS_DATA, MINE_RATE } = require("./config")
const { cryptoHash } = require("crypto")
```

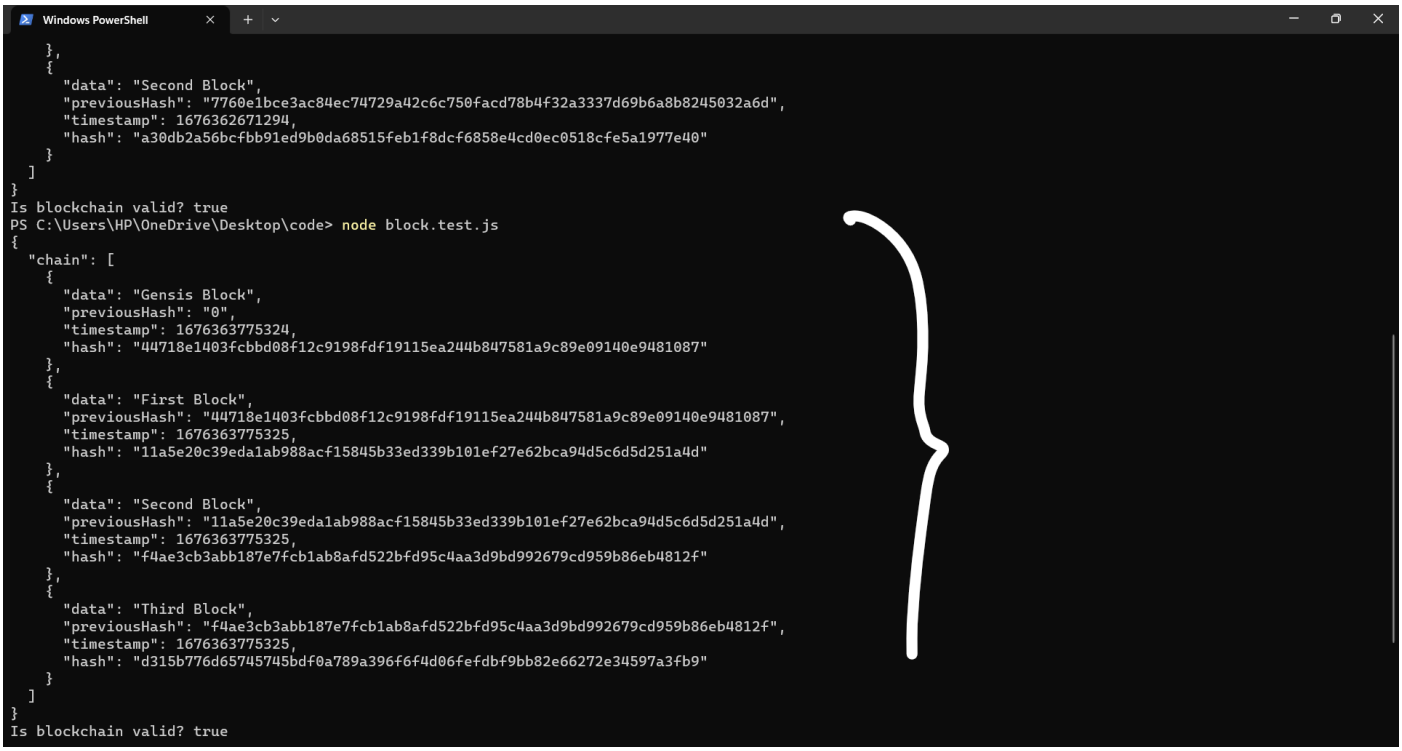
```
class Block {
  constructor({ timestamp, lastHash, hash, data, nonce, difficulty }) {
    this.timestamp = timestamp;
    this.lastHash = lastHash;
    this.hash = hash;
    this.data = data;
    this.nonce = nonce;
    this.difficulty = difficulty;
  }

  static gensis() {
    return new this(GENSIS_DATA);
  }

  static mineBlock({ lastBlock, data }) {
    const lastHash = lastBlock.hash;
    let hash, timestamp;
    let { difficulty } = lastBlock;
    let nonce = 0

    do {
      nonce++;
      timestamp = Date.noe()
      difficulty = Block.adjustDifficulty({ originalBlock: lastBlock, timestamp });
      hash = cryptoHash(timestamp, lastHash, data, nonce, difficulty)
    } while (hexToBinary(hash).substring(0, difficulty) !== '0'.repeat(difficulty))
    return new this({ timestamp, lastHash, data, difficulty, nonce, hash });
  }

  static adjustDifficulty({ originalBlock, timestamp }) {
    const { difficulty } = originalBlock;

    if (difficulty < 1) {
      return 1;
    }
```

```
        if ((timestamp - originalBlock.timestamp) > MINE_RATE) {
            return difficulty - 1;
        }

        return difficulty + 1;
    }
}

module.exports = Block;
```

Result/ Output/ Screenshots of the Practical :







Parameters achieved/ Conclusion :

Therefore, understood – implemented and created a blockchain with 3 blocks and check the validity of blockchain using SHA256.