Development and Analysis of a Blockchain System using JavaScript

A project report submitted in fulfilment of the

requirements for the degree of

Bachelor of Engineering

in

Computer Engineering

by
Brendan Lucas (8953)
Candida Noronha (8960)

Under the guidance of
Prof. Monica Khanore

Department Of Computer Engineering
Fr. Conceicao Rodrigues College Of Engineering
Fr. Agnel Ashram, Bandra (W),
Mumbai - 400 050.

UNIVERSITY OF MUMBAI
(2022 – 2023)

<u>CERTIFICATE</u>

This is to certify that the following students working on the project ***"Development and Analysis of a Blockchain System using JavaScript"*** *have satisfactorily completed the requirements of the project in fulfillment of the course T.E in Computer Engineering of the University of Mumbai during academic year 2022-2023 under the guidance of "Prof. Monica Khanore".*

Submitted By: ***Brendan Lucas (8953)***
Candida Noronha (8960)

Prof. Monica Kanore	**Dr. Sujata Deshmukh**
Teacher Incharge	**Head of the Department**

Principal

CERTIFICATE

*This is to certify that the project synopsis entitled **“Development and Analysis of a Blockchain System using JavaScript”** submitted by the following students is found to be satisfactory and the report has been approved as it satisfies the academic requirements in respect of mini-project work prescribed for the course.*

Submitted By: ***Brendan Lucas (8953)***
Candida Noronha (8960)

Internal Examiner	**External Examiner**
(Signature)	(Signature)
Name:	Name:
Date:	Date:

Seal of the Institute

DECLARATION OF THE STUDENT

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources.

We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in my submission.

We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

[illegible]

Date: 10-10-2022

Brendan Lucas
8953

Candida

Date: 10-10-2022

Candida Noronha
8960

ABSTRACT

Cryptocurrency is a new kind of money used for purchasing goods and services via payment system that uses peer-to-peer (P2P) technology. The transactions of these currencies are cryptographically verified as well as validated without any centralized authority to monitor or regulate the money value and production of additional units of money. Our project describes the main features of blockchain technology and provides the main areas of use of this technology. Today, blockchain has developed into a trustworthy and secure way to store and record transactions across many application domains. However, the impact of existing solutions on the current network infrastructure needs to be considered. We propose to create a system for generating and storing blockchain traffic based on the JavaScript programming language. This solution cannot be used with cryptocurrencies, but it does let you assess how well a distributed database built on blockchain technology works. Analysis of traffic generation and distribution delays as well as the relationship between computer performance and block generation and distribution speed are the main goals of our work. The algorithm for adding a new node to the cluster and adding a new block to the blockchain, as well as the design of a node in a blockchain cluster, were also taken into consideration in the project. Our project explains various steps involved in developing your own blockchain using JavaScript which is validated using the test-driven development (TDD) approach, thereby end up with a developer friendly API.

TABLE OF CONTENTS

1. INTRODUCTION

Blockchain is a distributed database that consists of an ever-growing list of structured data, and data storage and processing devices in this system are not connected to a common server. Blockchain is a network of interconnected blocks that adhere to certain principles. Each block contains transaction information as well as additional information, such as information about the previous block. Because of the system's design, it's difficult to make changes to existing blocks because you'll have to change the entire subsequent chain. Furthermore, blockchain is a decentralised system. This means that, unlike traditional centralised systems, the network does not rely on a single trusted authority to manage the system. Instead, trust unfolds as an emergent property as a result of network node interactions. At the same time, copies of chains are stored independently on numerous computers and servers, which adds to the difficulty of forging blocks. Let us examine the benefits and drawbacks of this technology.

Advantages include decentralisation, dependability and confidentiality, consensus, and system and operational transparency.

Disadvantages include scalability, fraud, and the irreversibility of operations in the event of an error; transaction processing speed is slower than in current systems; and the possibility of illegal or shady operations.

Initially, the term "blockchain" was applied to the database of the Bitcoin system; as a result, the blockchain is frequently referred to only for transactions in cryptocurrencies, of which there are many today. However, blockchain can be used in a variety of applications.

2. LITERATURE REVIEW

1. Blockchain has shown its potential for transforming traditional industry with its key characteristics: decentralization, persistency, anonymity and auditability. In this paper, they present a comprehensive overview on blockchain. They first gave an overview of blockchain technologies including blockchain architecture and key characteristics of blockchain. They then discuss the typical consensus algorithms used in blockchain. They analyzed and compared these protocols in different respects. Furthermore, they listed some challenges and problems that would hinder blockchain development and summarized some existing approaches for solving these problems. Some possible future directions are also proposed. Nowadays blockchain based applications are springing up and they plan to conduct in-depth investigations on blockchain-based applications in the future. [1]

2. Information technology has become a critical innovation in almost every industry. Those institutions or teams that can use technology correctly and effectively play a major role in disrupting the status quo in a leadership position. Those that don’t keep up with technology generally do not survive. The authors of this paper have identified the Blockchain technology as a catalyst for emerging use cases in the financial and nonfinancial industries such as industrial manufacturing, supply chain, and healthcare. The research indicates Blockchain can play a pivotal role in transforming the digitization of industries and applications by enabling secure trust frameworks, creating agile value chain production, and tighter integration with technologies such as cloud computing, and IoT. In producing a cloud-based application called HealthChain, the researchers have demonstrated the capability to apply professional engineering principles, combined with a DevOps approach to iterative development and management, and integration of cyber security, distributed computing, and Block-chain technologies. They feel HealthChain is one of many examples that demonstrate the transformative capability of Blockchain. Industry is looking to produce efficiencies, create new innovative products, and strengthen customer relationships globally by the effective use of mobile, IoT (Internet of Things), social media,

analytics and cloud technology to generate models for better decisions. Blockchain offers a secure way to exchange any kind of good, service, or transaction. Establishing the initial technology in the financial sector as given us insight and recommendations to be applied to other industries including health care where security, transformation and regulation plays a major role in advancing. Blockchain will enable more agile value chains, faster product innovations, closer customer relationships, and faster integration with the Internet of Things (IoT) and cloud technology. Blockchain allows immediate contracts, engagements, and agreements with inherent, robust cyber security features. This paper presented how Blockchain is changing the world. [2]

3. OBJECTIVES

- To study about how a blockchain can be deployed in different applications
- To implement a core blockchain i.e., its architecture, its working flow, and characteristics.

4. PROJECT DESCRIPTION

4.1 Overview of the project

Blockchain came as a solution to the longstanding user's trust problem. With its emergence with the renowned cryptocurrency, it provided an architecture to allow the user to trust a decentralized system instead of trusting a third party. Operating a peer-to-peer network, it keeps records of the ledger of transactions. This helps to avoid any center party. The whole process is done through a consensus. A ledger is shared between multiple entities, allowing everyone to inspect it. No single user can control it. It is a distributed cryptographically secured database that keeps the record of every transaction from the very initial one.

The following are the main features of our blockchain:

a. **Decentralized Computation**: Blockchain consists of distributed ledgers maintained by peer to peer networks . Blockchain eliminates the role of the central entity by using consensus protocol to validate transactions .

b. **Distributed Ledger of Transactions**: A shared ledger is used to store transactions. A copy of the ledger is maintained on every peer of the blockchain network. These copies are synchronized by timely replication .

c. **Transparency**: Blockchain stores every transaction in a block. Also, it is available to all the peers for verification .

d. **Security**: Each block is added to the chain after validation. Also, each block contains a hash of the previous block. It is computationally impossible to delete or update a block because it requires re-calculation of all the preceding block hashes.

e. **Fault Tolerant Network**: Blockchain has a peer-to-peer network of nodes. All miner nodes process transactions in parallel. The blockchain will continue working if some of the nodes fail to function.

4.2 Design and Methodology of proposed system

The information in a blockchain is stored in cryptographically encrypted chunks known as blocks. The next successive block contains information about the previous block and hence forms a chain. Thereby it gets its name. Each block in a blockchain contains a unique hash, transaction data and hash of the previous block. The initial block is known as genesis block. A genesis block does not contain a previous hash. Participants of the blockchain network can be organizations, individuals or institutions which share a copy of the ledger that contains their valid transactions in a sequential manner. The new transactions are added to the existing records by consensus of the miners participating in that network. To validate the transactions, miners have to implement the blockchain’s algorithm in order to be rewarded with a native token as per existing economic consensus mechanisms like proof of work, proof of stake, etc.

The fastest miner validates each transaction in a block and add it to the blockchain. A miner is selected from a pool of miners using a proof-of-work (PoW) consensus mechanism. The mining difficulty can be changed. A blockchain uses a consensus mechanism to allow the miners to agree on a single value. After successful validation by all the miners in the blockchain network, the block is added to the blockchain. The miner obtains a transaction fee and new block addition fee in case of PoW. The ledger runs on a peer-to-peer network and thus all the nodes participating in the network get a copy of the original information.

5. IMPLEMENTATION DETAILS

5.1 Hardware & Software Requirements

➔ Hardware Requirements

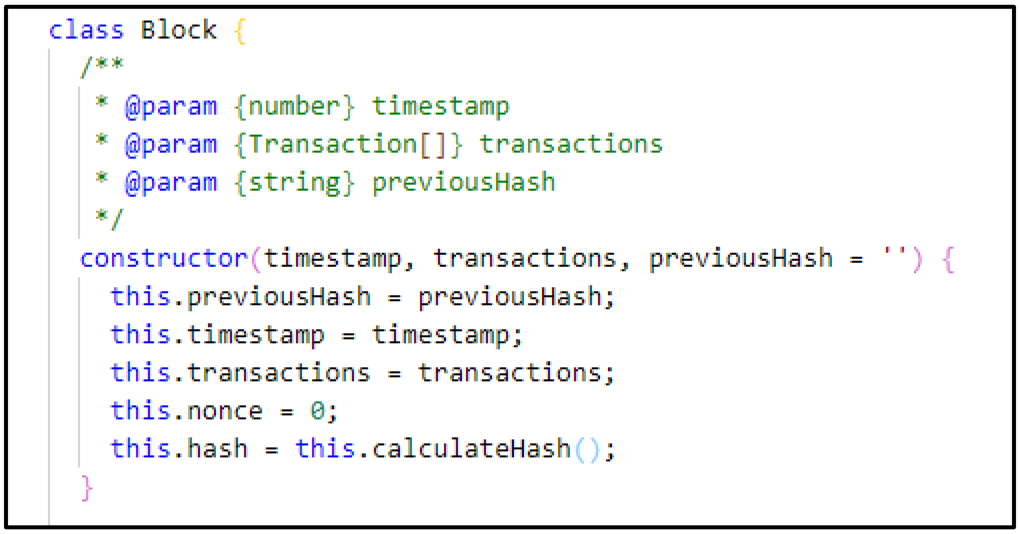
- Processor: Intel Core i3/Pentium or any advanced version
- Speed: 1.1 GHz (min)
- RAM: 4 GB (min)

➔ Software Requirements

- Operating System: Any version of Windows family (4.0 & above)
- Language: JavaScript
- Frontend: Angular

5.2 Implementation and Results

Block Structure:



```
class Block {
  /**
   * @param {number} timestamp
   * @param {Transaction[]} transactions
   * @param {string} previousHash
   */
  constructor(timestamp, transactions, previousHash = '') {
    this.previousHash = previousHash;
    this.timestamp = timestamp;
    this.transactions = transactions;
    this.nonce = 0;
    this.hash = this.calculateHash();
  }
```

We created the Block class and added the constructor() method to it — just like it's done in any other JavaScript class. Then, to initialize its properties, the following parameters are assigned to the constructor method:

Nonce: It's a unique number that tracks the position of every block in the entire blockchain.

Timestamp: It keeps a record of the time of occurrence of each completed transaction.

Transactions: It provides data about the completed transactions, such as the sender details, recipient's details, and quantity transacted.

Previous Hash: It points to the hash of the preceding block in the blockchain, something important in maintaining the blockchain's integrity.

Hash: Hash of the current Block.

Creating a Blockchain:

The Blockchain class will be responsible for handling the operations of the entire chain.

Code for the Blockchain class:

```
class Blockchain {
  constructor() {
    this.chain = [this.createGenesisBlock()];
    this.difficulty = 2;
    this.pendingTransactions = [];
    this.miningReward = 100;
  }
  /**
   * @returns {Block}
   */
  createGenesisBlock() {
    return new Block(Date.parse('2022-04-18'), [], '0');
  }

  /**
   * Returns the latest block on our chain. Useful when you want to create a
   * new Block and you need the hash of the previous Block.
   *
   * @returns {Block[]}
   */
  getLatestBlock() {
    return this.chain[this.chain.length - 1];
  }

  /**
   * Takes all the pending transactions, puts them in a Block and starts the
   * mining process. It also adds a transaction to send the mining reward to
   * the given address.
   *
   * @param {string} miningRewardAddress
   */
  minePendingTransactions(miningRewardAddress) {
    const rewardTx = new Transaction(
      null,
      miningRewardAddress,
      this.miningReward
    );
    this.pendingTransactions.push(rewardTx);

    const block = new Block(
      Date.now(),
      this.pendingTransactions,
      this.getLatestBlock().hash
    );
    block.mineBlock(this.difficulty);// mine a block with the difficulty that is
set within the blockchain

    debug('Block successfully mined!');
    this.chain.push(block);

    this.pendingTransactions = [];// reset the pending transactions
  }

  /**
   * Add a new transaction to the list of pending transactions (to be added
   * next time the mining process starts). This verifies that the given
   * transaction is properly signed.
   *
   * @param {Transaction} transaction
   */
  addTransaction(transaction) {
    if (!transaction.fromAddress || !transaction.toAddress) {
      throw new Error('Transaction must include from and to address');
    }
    // Verify the transactiion
    if (!transaction.isValid()) {
      throw new Error('Cannot add invalid transaction to chain');
    }

    if (transaction.amount <= 0) {
      throw new Error('Transaction amount should be higher than 0');
    }

    // Making sure that the amount sent is not greater than existing balance
    const walletBalance = this.getBalanceOfAddress(transaction.fromAddress);
    if (walletBalance < transaction.amount) {
      throw new Error('Not enough balance');
    }

    // Get all other pending transactions for the "from" wallet
    const pendingTxForWallet = this.pendingTransactions.filter(
      tx => tx.fromAddress === transaction.fromAddress
    );

    // If the wallet has more pending transactions, calculate the total amount
    // of spend coins so far. If this exceeds the balance, we refuse to add this
    // transaction.
    if (pendingTxForWallet.length > 0) {
      const totalPendingAmount = pendingTxForWallet
        .map(tx => tx.amount)
        .reduce((prev, curr) => prev + curr);

      const totalAmount = totalPendingAmount + transaction.amount;
      if (totalAmount > walletBalance) {
        throw new Error(
          'Pending transactions for this wallet is higher than its balance.'
        );
      }
    }

    this.pendingTransactions.push(transaction);
    debug('transaction added: %s', transaction);
  }

  /**
   * Returns the balance of a given wallet address.
   *
   * @param {string} address
```

```
     * @returns {number} The balance of the wallet
     */
    getBalanceOfAddress(address) {
      let balance = 0;

      // loop over all blocks of the blockchain
      for (const block of this.chain) {
        for (const trans of block.transactions) {
          if (trans.fromAddress === address) {
            balance -= trans.amount;
          }

          if (trans.toAddress === address) {
            balance += trans.amount;
          }
        }
      }

      debug('getBalanceOfAddress: %s', balance);
      return balance;
    }
    /**
     * Returns a list of all transactions that happened
     * to and from the given wallet address.
     *
     * @param  {string} address
     * @return {Transaction[]}
     */
    getAllTransactionsForWallet(address) {
      const txs = [];

      for (const block of this.chain) {
        for (const tx of block.transactions) {
          if (tx.fromAddress === address || tx.toAddress === address) {
            txs.push(tx);
          }
        }
      }

      debug('get transactions for wallet count: %s', txs.length);
      return txs;
    }

    /**
```

```
     * Loops over all the blocks in the chain and verify if they are properly
     * linked together and nobody has tampered with the hashes. By checking
     * the blocks it also verifies the (signed) transactions inside of them.
     *
     * @returns {boolean}
     */
    isChainValid() {
      // Check if the Genesis block hasn't been tampered with by comparing
      // the output of createGenesisBlock with the first block on our chain
      const realGenesis = JSON.stringify(this.createGenesisBlock());

      if (realGenesis !== JSON.stringify(this.chain[0])) {
        return false;
      }

      // Check the remaining blocks on the chain to see if there hashes and
      // signatures are correct
      for (let i = 1; i < this.chain.length; i++) {
        const currentBlock = this.chain[i];//ith position in the chain
        const previousBlock = this.chain[i - 1];
        if (previousBlock.hash !== currentBlock.previousHash) {
          return false;
        }

        if (!currentBlock.hasValidTransactions()) {
          return false;
        }

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

      return true;
    }
  }
```

Describing the roles of the helper methods that constitute the Blockchain class.

a. Constructor Method

This method instantiates the blockchain. Inside the constructor, the blockchain property is created, which refers to an array of blocks. The createGenesisBlock() method, is passed to it which creates the initial block in the chain.
We imported the crypto-js JavaScript library and used its crypto-js/sha256 module to calculate the hash of each block. Since the module returns a number object, we used the toString() method to convert it into a string.

b. Creating The Genesis Block

In a blockchain, the genesis block refers to the first-ever block created on the network. Whenever a block is integrated with the rest of the chain, it should reference the preceding block. Conversely, in the case of this initial block, it does not have any preceding block to point to. Therefore, a genesis block is usually hardcoded into the blockchain. This way, subsequent blocks can be created on it. It usually has an index of 0.
The createGenesisBlock() method is used to create the genesis block. It iscreated it using the Block class and the timestamp, transactions, and previousHash are passed to it.

c. Obtaining The Latest Block

Getting the latest block in the blockchain assists in ensuring the hash of the current block points to the hash of the previous block — thus maintaining the chain's integrity.
The getLatestBlock() method is used to retrieve it.

d. Adding New Blocks

The addTransaction() method is used to add a new block to the chain. To accomplish this, we set the previous hash of the new block to be equal to the hash of the last block in the chain — thus ensuring the chain is tamper-proof.
Since the properties of the new block get changed with every new calculation, it's important to calculate its cryptographic hash again. After updating its hash, the new block is pushed into the blockchain array.
In reality, adding a new block to a blockchain is not that easy because of the several checks that have been placed. Nonetheless, for this simple cryptocurrency, it's enough to demonstrate how a blockchain actually works.

Testing The Blockchain

```
'use strict';
const { Blockchain, Transaction } = require('./blockchain');
const EC = require('elliptic').ec;
const ec = new EC('secp256k1');
```

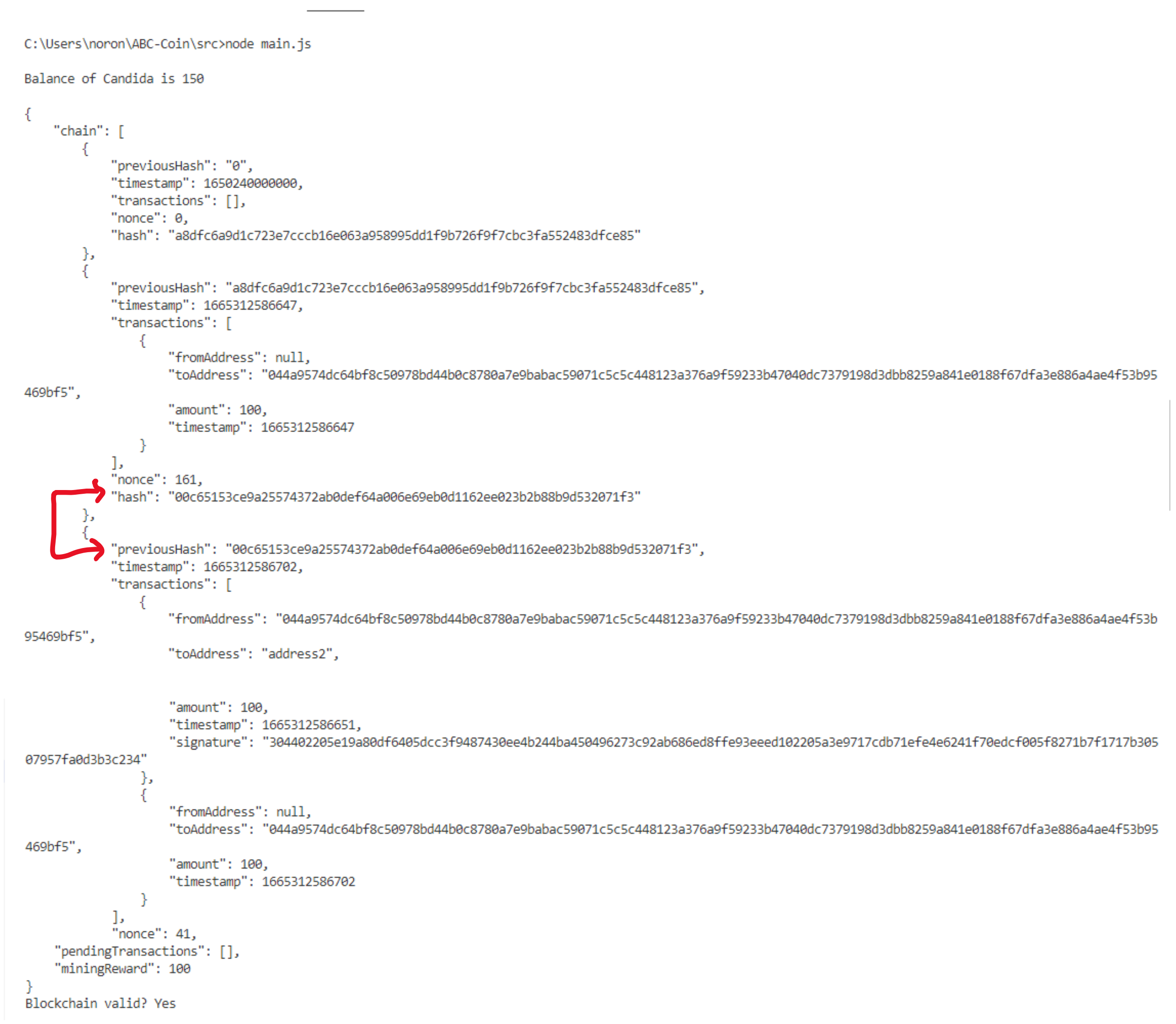
```
// Your private key goes here
const myKey = ec.keyFromPrivate(
  'fc07becb342e2c07f89d4d8327be1420c3faa7ec2097f74d4f2e8ca7a127258a'
);
// private key - 7c4c45907dec40c91bab3480c39032e90049f1a44f3e18c3e07c23e3273995cf
// From that we can calculate your public key (which doubles as your wallet
address)
const myWalletAddress = myKey.getPublic('hex');

// Create new instance of Blockchain class
const abcCoin = new Blockchain();
// Mine first block
abcCoin.minePendingTransactions(myWalletAddress);

// Create a transaction & sign it with your key
const tx1 = new Transaction(myWalletAddress, 'address2', 100);
tx1.signTransaction(myKey);
abcCoin.addTransaction(tx1);

// Mine block
abcCoin.minePendingTransactions(myWalletAddress);

// Create second transaction
const tx2 = new Transaction(myWalletAddress, 'address1', 50);
tx2.signTransaction(myKey);
abcCoin.addTransaction(tx2);
// Mine block
abcCoin.minePendingTransactions(myWalletAddress);
console.log();
console.log(
  `Balance of Candida is ${abcCoin.getBalanceOfAddress(myWalletAddress)}`
);
```



```
C:\Users\noron\ABC-Coin\src>node main.js

Balance of Candida is 150

{
    "chain": [
        {
            "previousHash": "0",
            "timestamp": 1650240000000,
            "transactions": [],
            "nonce": 0,
            "hash": "a8dfc6a9d1c723e7cccb16e063a958995dd1f9b726f9f7cbc3fa552483dfce85"
        },
        {
            "previousHash": "a8dfc6a9d1c723e7cccb16e063a958995dd1f9b726f9f7cbc3fa552483dfce85",
            "timestamp": 1665312586647,
            "transactions": [
                {
                    "fromAddress": null,
                    "toAddress": "044a9574dc64bf8c50978bd44b0c8780a7e9babac59071c5c5c448123a376a9f59233b47040dc7379198d3dbb8259a841e0188f67dfa3e886a4ae4f53b95
469bf5",
                    "amount": 100,
                    "timestamp": 1665312586647
                }
            ],
            "nonce": 161,
            "hash": "00c65153ce9a25574372ab0def64a006e69eb0d1162ee023b2b88b9d532071f3"
        },
        {
            "previousHash": "00c65153ce9a25574372ab0def64a006e69eb0d1162ee023b2b88b9d532071f3",
            "timestamp": 1665312586702,
            "transactions": [
                {
                    "fromAddress": "044a9574dc64bf8c50978bd44b0c8780a7e9babac59071c5c5c448123a376a9f59233b47040dc7379198d3dbb8259a841e0188f67dfa3e886a4ae4f53b
95469bf5",
                    "toAddress": "address2",


                    "amount": 100,
                    "timestamp": 1665312586651,
                    "signature": "304402205e19a80df6405dcc3f9487430ee4b244ba450496273c92ab686ed8ffe93eeed102205a3e9717cdb71efe4e6241f70edcf005f8271b7f1717b305
07957fa0d3b3c234"
                },
                {
                    "fromAddress": null,
                    "toAddress": "044a9574dc64bf8c50978bd44b0c8780a7e9babac59071c5c5c448123a376a9f59233b47040dc7379198d3dbb8259a841e0188f67dfa3e886a4ae4f53b95
469bf5",
                    "amount": 100,
                    "timestamp": 1665312586702
                }
            ],
            "nonce": 41,
    "pendingTransactions": [],
    "miningReward": 100
}
Blockchain valid? Yes
```

It’s an object that contains the blockchain property, which is an array containing all the blocks in the chain. From the image above, each block references the hash of the previous block. For example, the second block references the hash of the first block.

Verifying The Blockchain’s Integrity

As earlier mentioned, a key characteristic of a blockchain is that once a block has been added to the chain, it cannot be changed without invalidating the integrity of the rest of the chain. Therefore, to verify the integrity of the blockchain, We added a isChainValid() method to the Blockchain class.

Hashes are critical for ensuring the validity and security of a blockchain because any change in the contents of a block will result in the production of an entirely new hash, and invalidating the blockchain.
As such, the isChainValid() method will make use of if statements to verify whether the hash of every block has been tampered with. Starting from the first created block, it'll loop over the entire blockchain and check for its validity. Note that since the genesis block was hardcoded, it'll not be checked.
Also, the method will verify whether the hashes of each two consecutive blocks are pointing to one another. If the integrity of the blockchain has not been compromised, it returns true; otherwise, in case of any anomalies, it returns false.

Code:

```
isChainValid() {
    // Check if the Genesis block hasn't been tampered with by comparing
    // the output of createGenesisBlock with the first block on our chain
    const realGenesis = JSON.stringify(this.createGenesisBlock());

    if (realGenesis !== JSON.stringify(this.chain[0])) {
      return false;
    }
    // Check the remaining blocks on the chain to see if there hashes and
    // signatures are correct
    for (let i = 1; i < this.chain.length; i++) {
      const currentBlock = this.chain[i];// ith position in the chain
      const previousBlock = this.chain[i - 1];
      if (previousBlock.hash !== currentBlock.previousHash) {
        return false;
      }
      if (!currentBlock.hasValidTransactions()) {
        return false;
      }
      if (currentBlock.hash !== currentBlock.calculateHash()) {
        return false;
      }
    }

    return true;
  }
```

Adding Proof-Of-Work

Proof of work is the concept applied to increase the difficulty entailed in mining or adding new blocks to the blockchain.
In the case of ABC Coin, we employed a simple algorithm that deters people from generating new blocks easily or spamming the blockchain.
So, in the Block class, we added another method called mineBlock().Essentially, this simple algorithm identifies a number, passed as a difficulty property, such that the hash of every block contains leading zeros that correspond to this difficulty level.
Ensuring the hash of every block begins with the number of zeros as set in the difficulty level requires a lot of computing power. The higher the difficulty level, the more time it takes to mine new blocks.
Furthermore, we added a random nonce value to every hashed block such that, when rehashing takes place, the difficulty level restrictions can still be met.

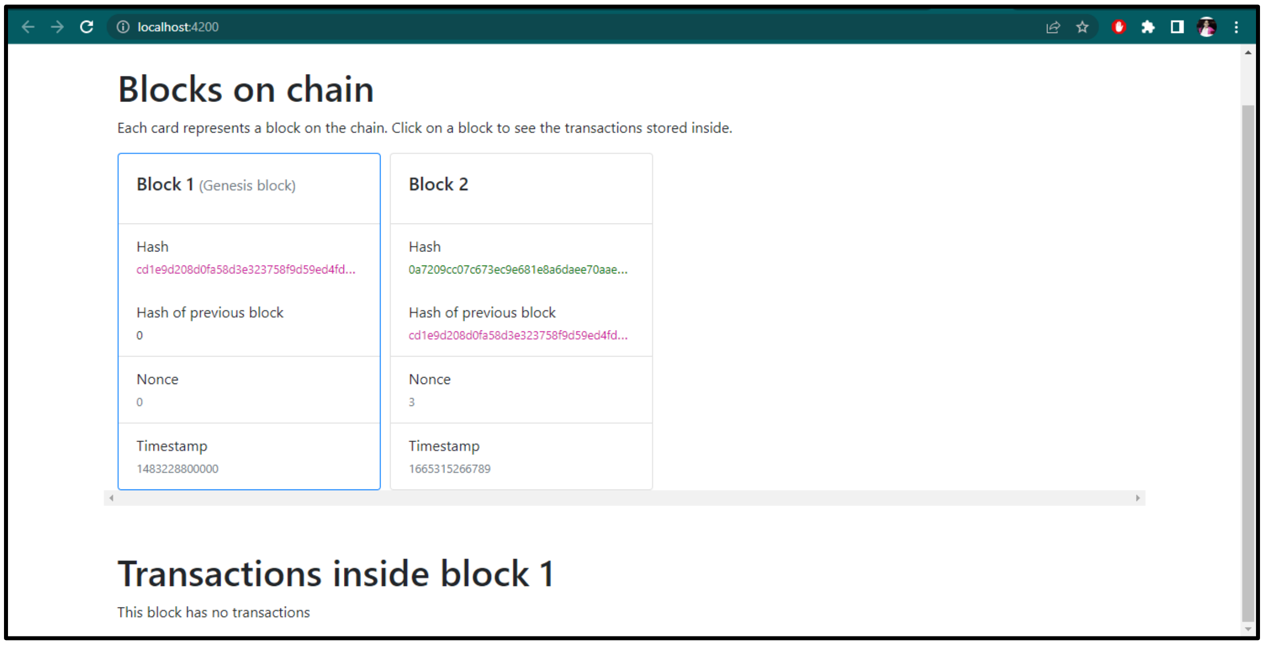
```
mineBlock(difficulty) {
    // if difficulty is set to 5 it will take first 5 characters of the hash and
check if all 5 are zeros

    // creating a string of 0 of length difficulty
    while (
      this.hash.substring(0, difficulty) !== Array(difficulty + 1).join('0')
    ) {
      this.nonce++;
      this.hash = this.calculateHash();
    }

    debug(`Block mined: ${this.hash}`);
  }
```

Viewing the Blockchain Functionality on the Front-end:

Genesis Block:



localhost:4200

Blocks on chain

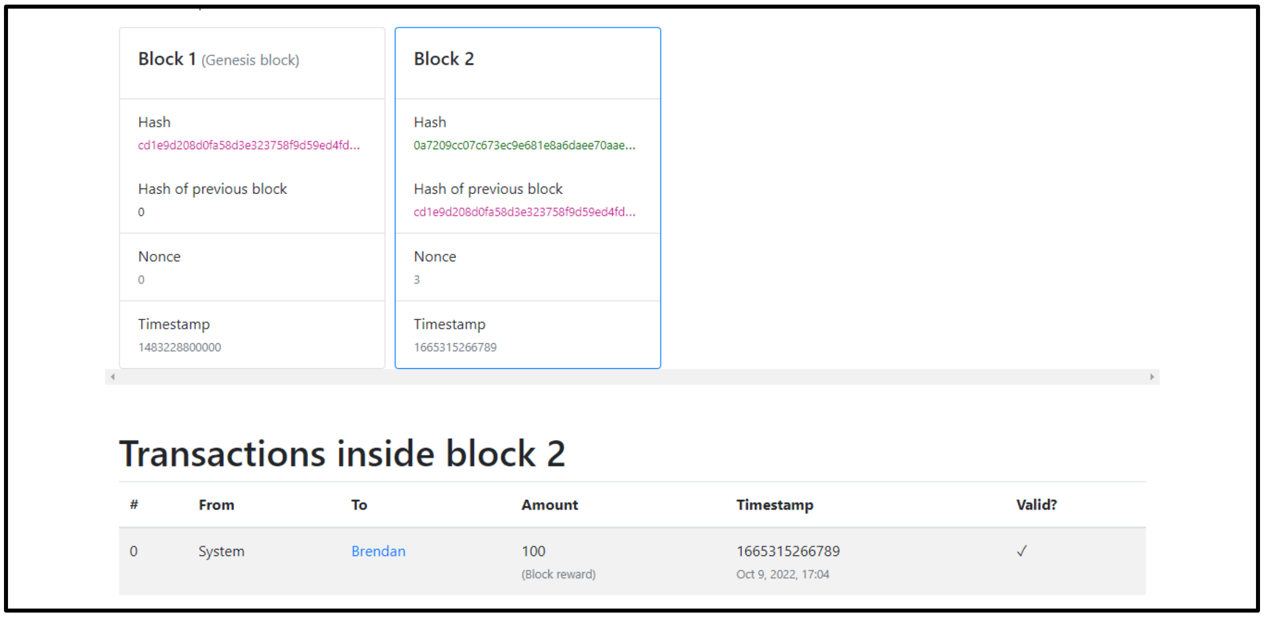
Each card represents a block on the chain. Click on a block to see the transactions stored inside.

Block 1 (Genesis block)	Block 2
Hash cd1e9d208d0fa58d3e323758f9d59ed4fd...	Hash 0a7209cc07c673ec9e681e8a6daee70aae...
Hash of previous block 0	Hash of previous block cd1e9d208d0fa58d3e323758f9d59ed4fd...
Nonce 0	Nonce 3
Timestamp 1483228800000	Timestamp 1665315266789

Transactions inside block 1

This block has no transactions

Block 2 Transactions:

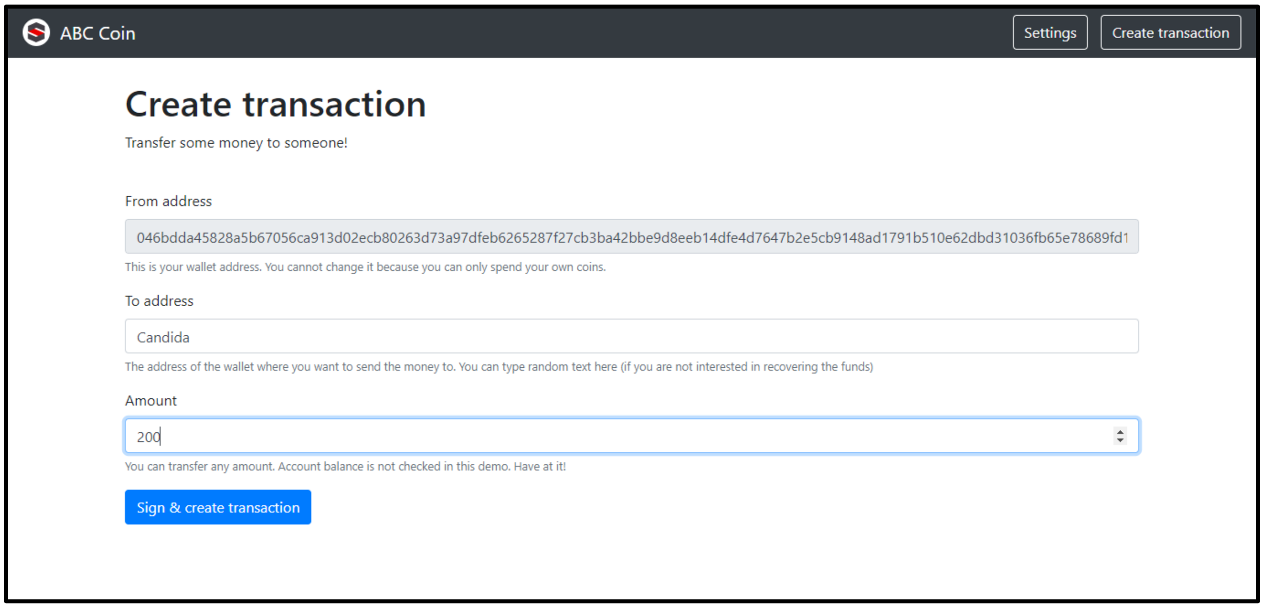


Block 1 (Genesis block)	Block 2
Hash cd1e9d208d0fa58d3e323758f9d59ed4fd...	Hash 0a7209cc07c673ec9e681e8a6daee70aae...
Hash of previous block 0	Hash of previous block cd1e9d208d0fa58d3e323758f9d59ed4fd...
Nonce 0	Nonce 3
Timestamp 1483228800000	Timestamp 1665315266789

Transactions inside block 2

#	From	To	Amount	Timestamp	Valid?
0	System	Brendan	100 (Block reward)	1665315266789 Oct 9, 2022, 17:04	✓

Create a new Transaction:



ABC Coin Settings Create transaction

Create transaction

Transfer some money to someone!

From address

046bdda45828a5b67056ca913d02ecb80263d73a97dfeb6265287f27cb3ba42bbe9d8eeb14dfe4d7647b2e5cb9148ad1791b510e62dbd31036fb65e78689fd1

This is your wallet address. You cannot change it because you can only spend your own coins.

To address

Candida

The address of the wallet where you want to send the money to. You can type random text here (if you are not interested in recovering the funds)

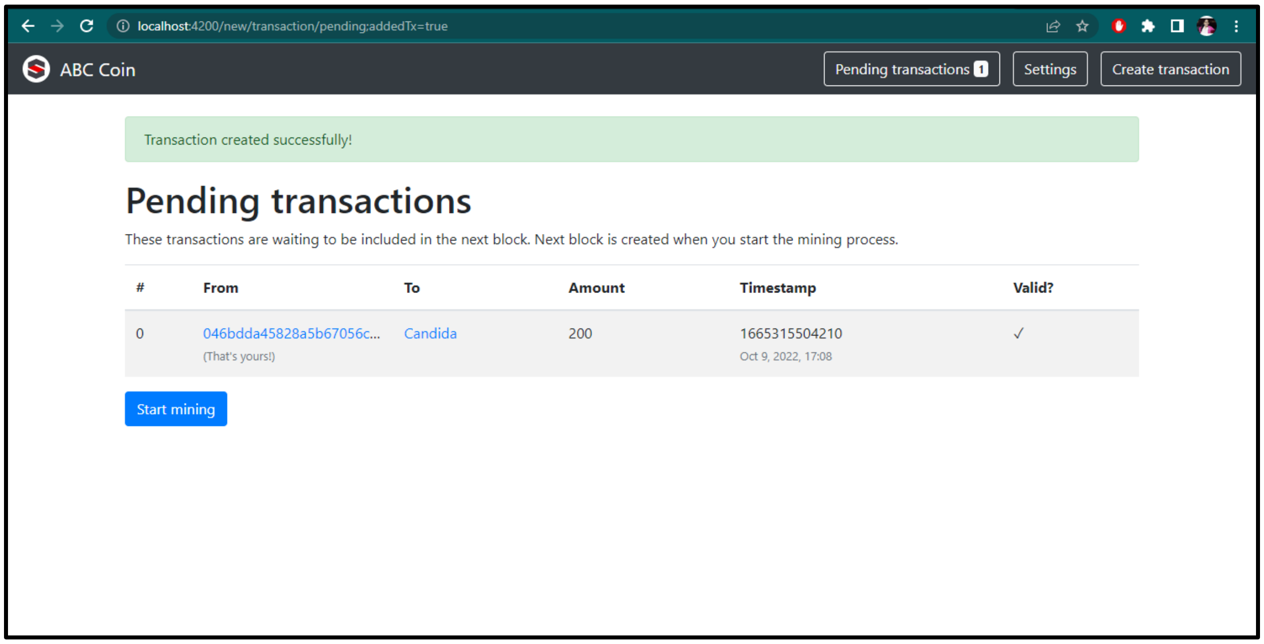
Amount

200

You can transfer any amount. Account balance is not checked in this demo. Have at it!

Sign & create transaction

Pending Transactions:



localhost:4200/new/transaction/pending;addedTx=true

ABC Coin Pending transactions 1 Settings Create transaction

Transaction created successfully!

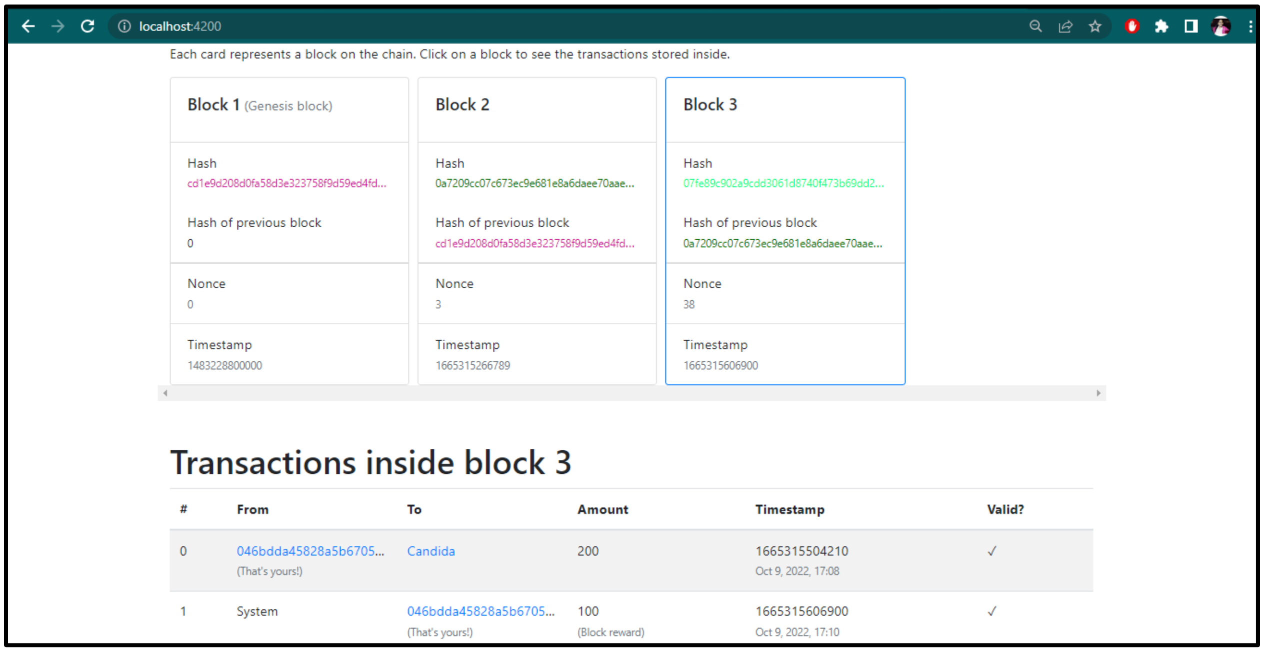
Pending transactions

These transactions are waiting to be included in the next block. Next block is created when you start the mining process.

#	From	To	Amount	Timestamp	Valid?
0	046bdda45828a5b67056c... (That's yours!)	Candida	200	1665315504210 Oct 9, 2022, 17:08	✓

Start mining

Newly Mined Block:



Each card represents a block on the chain. Click on a block to see the transactions stored inside.

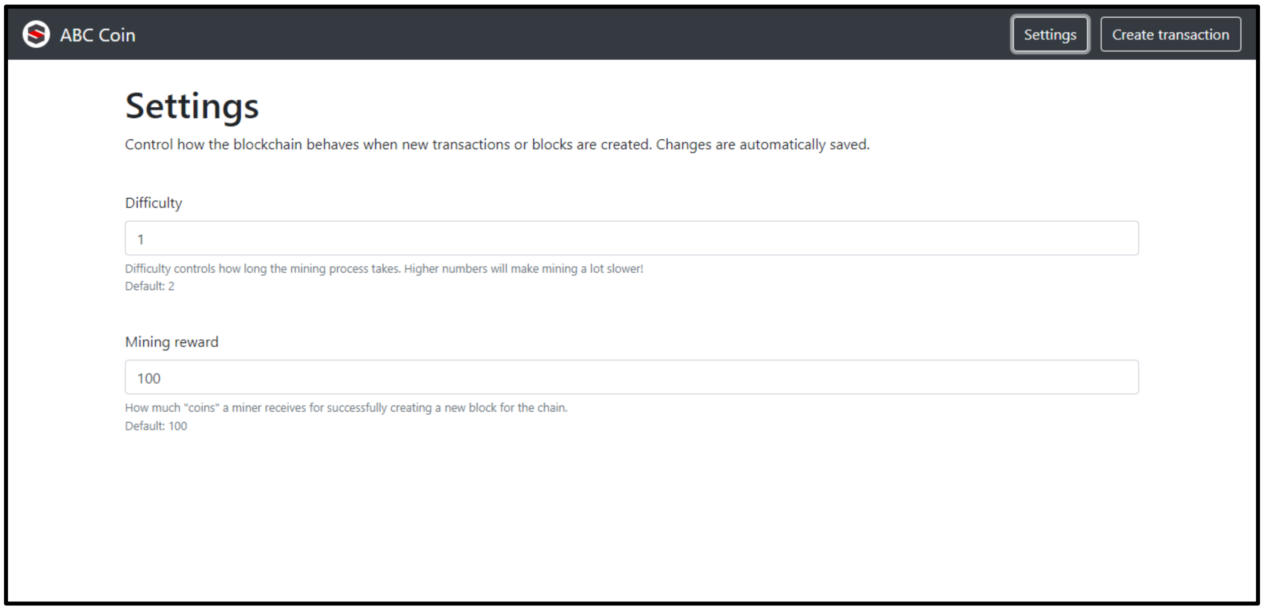
Block 1 (Genesis block)	Block 2	Block 3
Hash: cd1e9d208d0fa58d3e323758f9d59ed4fd...	Hash: 0a7209cc07c673ec9e681e8a6daee70aae...	Hash: 07fe89c902a9cdd3061d8740f473b69dd2...
Hash of previous block: 0	Hash of previous block: cd1e9d208d0fa58d3e323758f9d59ed4fd...	Hash of previous block: 0a7209cc07c673ec9e681e8a6daee70aae...
Nonce: 0	Nonce: 3	Nonce: 38
Timestamp: 1483228800000	Timestamp: 1665315266789	Timestamp: 1665315606900

Transactions inside block 3

#	From	To	Amount	Timestamp	Valid?
0	046bdda45828a5b6705... (That's yours!)	Candida	200	1665315504210 Oct 9, 2022, 17:08	✓
1	System	046bdda45828a5b6705... (That's yours!)	100 (Block reward)	1665315606900 Oct 9, 2022, 17:10	✓

Settings:

- **Change Mining Difficulty**
- **Change Mining Reward**



Settings

Control how the blockchain behaves when new transactions or blocks are created. Changes are automatically saved.

Difficulty

1

Difficulty controls how long the mining process takes. Higher numbers will make mining a lot slower!
Default: 2

Mining reward

100

How much "coins" a miner receives for successfully creating a new block for the chain.
Default: 100

6. CONCLUSION

The goal of the project was to study decentralized technology blockchain and its key elements which are smart contract as well as creating a cryptocurrency transfer application where the user can send a transaction with a rapid and simple process. The result was an Angular application that connects to the blockchain built using JavaScript. Each transaction sent by the user was permanently stored on the network. The application worked as a basic example to demonstrate the benefit of applying blockchain in providing transparency and trust as well as optimizing the time and cost in the transaction.

Overall, the project was successful as the goals that were stated at the beginning of this thesis were achieved. Based on this study, applying the concept of blockchain into an Angular application assisted effectively in consolidating the theories. For further developments, the application can be deployed to the domain as it is hosted on localhost now. Using this application, the user can use clearly understand the functionality of the Blockchain and can visualize the transactions.

7. REFERENCES

1. Z. Zheng, S. Xie, H. Dai, X. Chen and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," 2017 IEEE International Congress on Big Data (BigData Congress), 2017, pp. 557-564, doi: 10.1109/BigDataCongress.2017.85.

2. T. Ahram, A. Sargolzaei, S. Sargolzaei, J. Daniels and B. Amaba, "Blockchain technology innovations," 2017 IEEE Technology & Engineering Management Conference (TEMSCON), 2017, pp. 137-141, doi: 10.1109/TEMSCON.2017.7998367.

3. L. Carlozo (2017). What is blockchain? Journal of Accountancy. Vol. 224. No. 1. P. 29.

4. T. Ahram et al. (2017). Blockchain technology innovations. 2017 IEEE technology & engineering management conference (TEMSCON). IEEE. P. 137-141.

5. https://lisk.com/blog/research/blockchain-javascript-how-create-and-develop-blockchain-using-javascript

6. https://www.mdpi.com/2076-3417/11/14/6252/htm

7. https://www.smashingmagazine.com/2020/02/cryptocurrency-blockchain-node-js/