

A
Minor Project Report on
“IMPLEMENTATION OF BLOCKCHAIN USING
PYTHON”

In partial fulfillment of requirements for the degree of
Bachelor of Technology (B. Tech.)

in
Computer Science and Engineering



Submitted by
Ms. Oshi Jain (170316)

Under the Guidance of
Mr. Siddhant Kumar Singh

Computer Science and Engineering
SCHOOL OF ENGINEERING AND TECHNOLOGY
Mody University and Science and Technology
Lakshmangarh, Distt. Sikar-332311

December 2020

A C K N O W L E D G E M E N T

I sincerely express my gratitude to my guide Mr. Siddhant Kumar Singh for his benevolent guidance in completing the report on **“IMPLEMENTATION OF BLOCKCHAIN USING PYTHON”**. His kindness and help have been the source of encouragement for me.

I am grateful to him for the guidance, inspiration and constructive suggestions that were helpful in the preparation of this project.

Oshi Jain (170316)

CERTIFICATE

This is to certify that the minor project report entitled “ **IMPLEMENTATION OF BLOCKCHAIN USING PYTHON** ” submitted by Ms. Oshi Jain, as a partial fulfillment for the requirement of B. Tech. VII Semester examination of the School of Engineering and Technology, Mody University of Science and Technology, Lakshmangarh for the academic session 2019-2020 is an original project work carried out under the supervision and guidance of Mr. Siddhant Kumar Singh has undergone the requisite duration as prescribed by the institution for the project work.

PROJECT GUIDE:

Approval Code: AUT_20_CSE_F19_01

Name: Mr. Siddhant Kumar Singh

Date: December 27, 2020

HEAD OF DEPARTMENT

Signature:

Name:

Date:

EXAMINER-I:

Name: Dr. Ajay Kumar Singh

Dept: CSE

EXAMINER-II

Name: Dr. Sunil Kumar Jangir

Dept: CSE

ABSTRACT

Blockchain. What exactly is meant by blockchain ? Well blockchain in simple words we can say is a growing list of records or blocks which are linked with each other using cryptography. Inside a blockchain , each block contains a cryptographic hash of the previous block, transaction data and a timestamp.

By design, a blockchain is immutable or resistant to modification. That means nobody can easily tamper with the data. The data that is once recorded , cannot be changed or modified unless and until all the subsequent blocks are modified. As the records inside a blockchain are unalterable, it can be considered as secure by design.

A blockchain by nature is distributed, decentralized and most of the time is public. It is basically a digital ledger in which transactions which are validated at first , they are hashed and encoded into a merkle tree. They are then stored inside the blocks and the blocks inside a chain. And this linking of the blocks with one another is what makes a blockchain.

Inside a blockchain , since the data is stored in a decentralized manner , it simply eliminates number of risks that comes with data being kept or held in a centralized manner. For example, there is this whole network where we have a large number of computers and here in blockchain, each computer acts as a node. At every node we have a copy of blockchain. So suppose if somebody tries to tamper with the data at a particular node or computer, then the hash of that particular block will get changed and since other blocks are too linked with that block , the hashes of those blocks will also get changed. So we easily get to know that. So suppose if somebody tries to tamper with the data at a particular node or computer, then the hash of that particular block will get changed and since other blocks are too linked with that block , the hashes of those blocks will also get changed. So we easily get to know that yes somebody has tried to tamper with the data. So here, what we do exactly is that, since we have got copy of the blockchain on every single node, we simply copy the blockchain from any other node on the computer, on which alteration took place. So that's how we can easily identify inside a blockchain and that's what makes blockchain resistant to modification.

Table of Contents

Sr.no.	Topics	Page no.
1.	Introduction	1
1.1	Present System	2
1.2	Proposed System	2-3
2.	System Design	4
2.1	System Flowchart	4
3.	Hardware and Software Details	5
4.	Implementation Work Details	6
4.1	Real life Applications	6
4.2	Data implementation and Program Execution	7
5.	Source Code	8-15
6.	Input/Output Screens	16-17
7.	Conclusion	18
7.1	Limitations	18
7.2	Scope for Future Work	18-19
8.	Bibliography	20
9.	Annexures (Plagiarism Report)	21

List of Figures

<u>Sr. No.</u>	<u>Title</u>	<u>Page No.</u>
1.	Choice Set	16
2.	Displaying a Block	16
3.	Blockchain with 2 Blocks	17
4.	Blockchain with 2 Blocks	17

Chapter 1: Introduction

Blockchain. What exactly is meant by blockchain ? Well blockchain in simple words we can say is a growing list of records or blocks which are linked with each other using cryptography. Inside a blockchain , each block contains a cryptographic hash of the previous block, transaction data and a timestamp.

By design, a blockchain is immutable or resistant to modification. That means nobody can easily tamper with the data. The data that is once recorded , cannot be changed or modified unless and until all the subsequent blocks are modified. As the records inside a blockchain are unalterable, it can be considered as secure by design.

A blockchain by nature is distributed, decentralized and most of the time is public. It is basically a digital ledger in which transactions which are validated at first , they are hashed and encoded into a merkle tree. They are then stored inside the blocks and the blocks inside a chain. And this linking of the blocks with one another is what makes a blockchain.

Inside a blockchain , since the data is stored in a decentralized manner , it simply eliminates number of risks that comes with data being kept or held in a centralized manner. For example, there is this whole network where we have a large number of computers and here in blockchain, each computer acts as a node. At every node we have a copy of blockchain. So suppose if somebody tries to tamper with the data at a particular node or computer, then the hash of that particular block will get changed and since other blocks are too linked with that block , the hashes of those blocks will also get changed. So we easily get to know that. So suppose if somebody tries to tamper with the data at a particular node or computer, then the hash of that particular block will get changed and since other blocks are too linked with that block , the hashes of those blocks will also get changed. So we easily get to know that yes somebody has tried to tamper with the data. So here, what we do exactly is that, since we have got copy of the blockchain on every single node, we simply copy the blockchain from any other node on the computer, on which alteration took place. So that's how we can easily identify inside a blockchain and that's what makes blockchain resistant to modification.

1.1 PRESENT SYSTEM

Blockchain as such can be used in a very simple manner to a very advanced manner as well. What we get to see is that blockchain is a technology that underlies bitcoin. One of the biggest application of blockchain is bitcoin cryptocurrency. Here, bitcoin is the largest used cryptocurrency worldwide.

What present system here suggests is that blockchain as such is being implemented in a very advanced manner incorporating various other kind of features. It is at present is being used by large firms or tech giants as well such as IBM. Using blockchain we can also develop a blockchain transactions investigation tool as well. There are several other systems where they have developed or created interfaces using which we can interact with ethereum blockchain as well.

The main aim of the blockchains is to set up a creditworthy environment among independent participants in a non-trustable distributed environment. Another improvement or development in the present system of blockchain took place when facebook announced that the company is developing its own cryptocurrency i.e Libra and the DCEP by Central Bank of China.

The present system in itself is being developed in a very advanced and innovated manner as well. In spite of the fact that how bitcoin booms or downfalls, the blockchain technologies and applications are being continuously improved and innovated.

1.2 PROPOSED SYSTEM

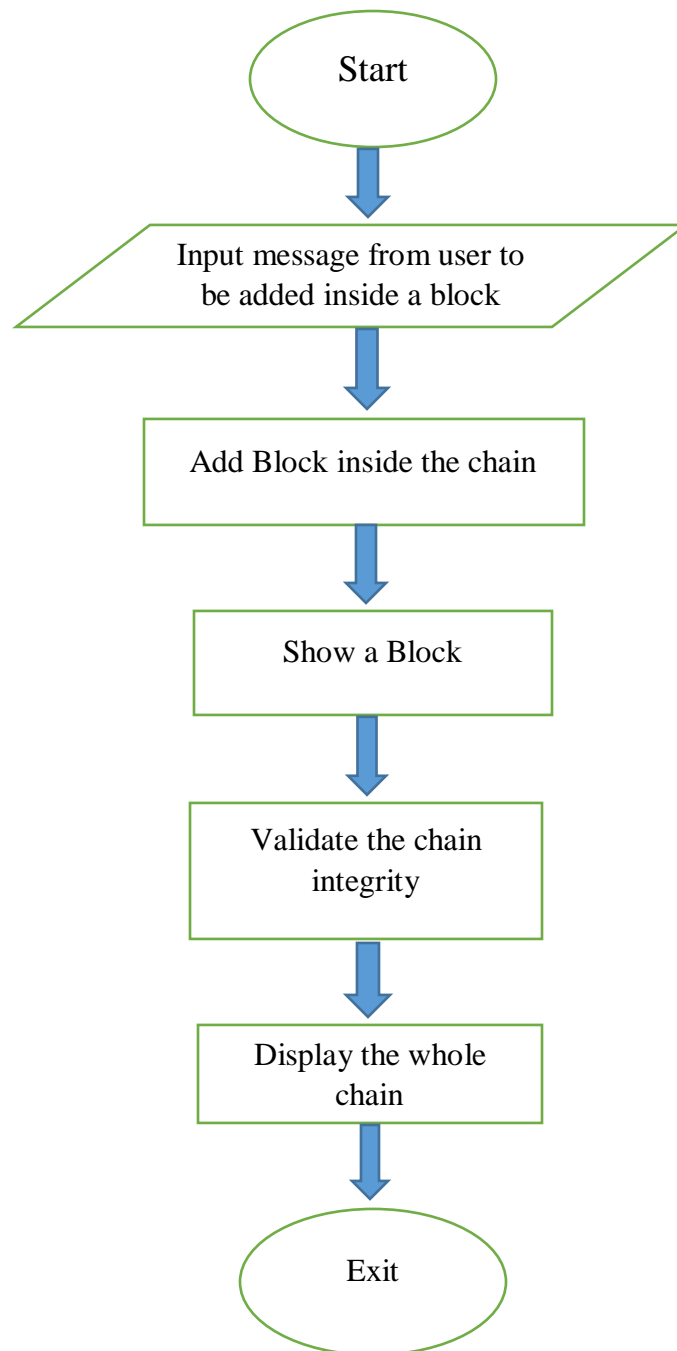
So, in my system or in my project what we have tried to do is implementing this technology or the main concept that lies behind this technology. What we have done in this project is a pretty simple, very easy to understand , an implementation of the blockchain technology. Here, inside our project we have not implemented very advanced kind of features like proof-of-work, distributed ledger or consensus protocol. And also in our project we have abstracted the idea of a transaction more to a general concept of message that can contain any type of data.

What exactly here we have done is we have decomposed the whole idea or the whole process of creating a blockchain. We have created three different classes for different operations i.e class message, class block and class chain. Each of the classes have got there own functionality. And also we have created a manager function so that we as user can interact with the blockchain easily.

The main intention or the main aim behind the development of this project is to explain and make clearer this idea that how exactly is blockchain is structured at its very core.

Chapter 2: System Design

2.1 System Flowchart



Chapter 3: Hardware and Software Details

3.1: Hardware Used:

- **Operating System:** Windows 10
- **RAM:** 8 GB
- **Processor:** i5 10th Generation Processor
- **Hard Disk:** 1 TB

3.2 Software Used:

The software used in this project is Spyder IDE. Spyder is basically an open-source IDE i.e integrated development environment designed with the purpose of scientific programming in python language. It was initially created and developed by Pierre Raybaut in 2009 and since 2012, it's being maintained and constantly improved by a team of scientific Python developers and therefore the community.

Spyder integrates with variety of prominent packages within the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, also as other open source software. It is released under the MIT license.

Spyder is actually available cross-platform through Anaconda, on Windows, on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian, Fedora, Gentoo Linux, openSUSE and Ubuntu. Some of its features are:

1. Editor with introspection, code completion
2. It supports multiple IPython consoles
3. Ability to explore and edit variables from a GUI
4. Static code analysis
5. A history log
6. An internal console

Chapter 4: Implementation Work Details

4.1 Real life applications

Blockchain as such in real life has got many applications. Some of them are:

1. Supply Chain Management

To increase the overall efficiency of supply chains we can use blockchain. Using blockchain we can accurately identify the location of items on supply chain. And in this way it helps in removing the paper work and also help in preventing losses.

2. Healthcare

Patient being the central point, has complete right to accurate information. And data of a patient needs to be kept in a private and secure manner so that nobody gets to tamper with the data easily. Hospitals now-a-days have moved from the traditional paper work and are storing the data on blockchain that is kept confidential. Using blockchain in this field, diagnosis of patient's can also be stored.

3. Wills or Inheritances

Now-a-days paper wills can be replaced with digital ones that can be created and stored using blockchain technology provided it should be used with smart contracts so as to make the documents both legally viable and crystal clear and it will also put our end of life concerns to the rest.

4. Digital Voting

Many a times what we get to see is that there is some fraud or malpractices being took place while voting. Because of the blockchain technology immutable nature, we can actually make our vote count. Using this technology, the mechanism of voting will be even more transparent and if any changes are made or something wrong took place, it can be easily tracked. The token-based system that is created using this technology will ensure one unchangeable vote per person.

4.2 Data Implementation and Program Execution

The execution of the program goes like this:

So at first when we run the program the user is provided with a set of choices like what exactly he wants to do at first. So the set of the choices which we have provided are:

1. Add message to the block
2. Add existing block to the chain
3. Show a block (here it will ask index too)
4. Validate the chain integrity
5. Display the whole chain
6. Exit

These are the set of the choices which we have provided. So first of all user needs to enter some message so that it can be added inside the block. If he wishes to add more than one messages then also he can do that.

After this you are supposed to add the existing block to the chain so that if you wish to see a block so at that time you can provide the index of the block which you want to be displayed.

What comes after this is before displaying the whole chain of blocks, we should validate the integrity of the chain. So for that we have provided the user an option to do so.

After validating the chain integrity, what comes is pretty simple. That means we are simply displaying the entire blockchain. Here when we display the entire blockchain , what we get to see is:

1. Hash of the previous block (if any)
2. Hash of the block itself
3. Number of messages inside a block
4. Timestamp

And the last what we have provided is the exit option if the user wishes to leave or stop the execution of the program. So these are the things which we will get to see and that's how the flow of the program or the execution of the program goes.

Chapter 5: Source Code

```
import datetime

import hashlib

import time

class Message:

    def __init__(self, data):

        self.hash = None

        self.prev_hash = None

        self.timestamp = time.time()

        self.size = len(data.encode('utf-8')) # length in bytes

        self.data = data

        self.payload_hash = self._hash_payload()

    def _hash_payload(self):

        return hashlib.sha256(bytearray(str(self.timestamp) + str(self.data), "utf-8")).hexdigest()

    def _hash_message(self):

        return hashlib.sha256(bytearray(str(self.prev_hash) + self.payload_hash, "utf-8")).hexdigest()

    def link(self, message):
```

```

        """ Links the message to the previous one via hashes. """

        self.prev_hash = message.hash

def seal(self):

    """ Get the message hash. """

    self.hash = self._hash_message()

def validate(self):

    """ Check whether the message is valid or not. """

    if self.payload_hash != self._hash_payload():

        raise InvalidMessage("Invalid payload hash in message: " + str(self))

    if self.hash != self._hash_message():

        raise InvalidMessage("Invalid message hash in message: " + str(self))

def __repr__(self):

    return 'Message<hash: {}, prev_hash: {}, data: {}>'.format(

        self.hash, self.prev_hash, self.data[:20]

    )

class Block:

    def __init__(self, *args):

        self.messages = []

        self.timestamp = None

        self.prev_hash = None

```

```

self.hash = None

if args:

    for arg in args:

        self.add_message(arg)

def _hash_block(self):

    return hashlib.sha256(bytearray(str(self.prev_hash) + str(self.timestamp) +
self.messages[-1].hash, "utf-8")).hexdigest()

def add_message(self, message):

    if len(self.messages) > 0:

        message.link(self.messages[-1])

    message.seal()

    message.validate()

    self.messages.append(message)

def link(self, block):

    """ The block hash only incorporates the head message hash

        which then transitively includes all prior hashes.

        """

    self.prev_hash = block.hash

def seal(self):

```



```

self.timestamp = time.time()

self.hash = self._hash_block()

def validate(self):
    """ Validates each message hash, then chain integrity, then the block hash.

        Calls each message's validate() method.

        If a message fails validation, this method captures the exception and
        throws InvalidBlock since an invalid message invalidates the whole block.
    """

    for i, msg in enumerate(self.messages):

        try:

            msg.validate()

            if i > 0 and msg.prev_hash != self.messages[i-1].hash:

                raise InvalidBlock("Invalid block: Message #{ } has invalid
message link in block: { }".format(i, str(self)))

        except InvalidMessage as ex:

            raise InvalidBlock("Invalid block: Message #{ } failed validation:
{ }. In block: { }".format(

                i, str(ex), str(self))

            )

    def __repr__(self):

        return 'Block<hash: { }, prev_hash: { }, messages: { }, time: { }>'.format(

```

```
        self.hash, self.prev_hash, len(self.messages), self.timestamp
    )
```

```
class SimpleChain:
```

```
    def __init__(self):
```

```
        self.chain = []
```

```
    def add_block(self, block):
```

```
        """ Add a block if valid."""
```

```
        if len(self.chain) > 0:
```

```
            block.prev_hash = self.chain[-1].hash
```

```
            block.seal()
```

```
            block.validate()
```

```
            self.chain.append(block)
```

```
    def validate(self):
```

```
        """ Validates each block, in order.
```

```
        An invalid block invalidates the chain.
```

```
        """
```

```
        for i, block in enumerate(self.chain):
```

```
            try:
```

```
                block.validate()
```

```
            except InvalidBlock as exc:
```

```
        raise InvalidBlockchain("Invalid blockchain at block number {}".format(i, str(exc)))
    caused by: {}".format(i, str(exc)))
```

```
    return True
```

```
def __repr__(self):
```

```
    return 'SimpleChain<blocks: {}>'.format(len(self.chain))
```

```
class InvalidMessage(Exception):
```

```
    def __init__(self, *args, **kwargs):
```

```
        Exception.__init__(self, *args, **kwargs)
```

```
class InvalidBlock(Exception):
```

```
    def __init__(self, *args, **kwargs):
```

```
        Exception.__init__(self, *args, **kwargs)
```

```
class InvalidBlockchain(Exception):
```

```
    def __init__(self, *args, **kwargs):
```

```
        Exception.__init__(self, *args, **kwargs)
```

```
def manager():
```

```
    chain = SimpleChain()
```

```
    block = Block()
```

```
    msg = ""
```

```
    Implementation of a Blockchain. Once recorded cannot be reversed.
```

Choice set:

- add message to the existing block (1)
- add existing block to the chain (2)
- show a block (index will be asked) (3)
- show the whole chain (4)
- validate the chain integrity (5)
- exit the program (6)

"""

```
print(msg)
```

```
while True:
```

```
    print()
```

```
    decide = input("Your action: ")
```

```
    if decide == "1":
```

```
        block.add_message(Message(input("Enter your data:")))
```

```
    elif decide == "2":
```

```
        if len(block.messages) > 0:
```

```
            chain.add_block(block)
```

```
            block = Block()
```

```
        else: print("Block is empty, try adding some messages")
```

```
    elif decide == "3":
```

```
        index = int(input("Provide the index: "))
```

```
        if len(chain.chain)>0:
```

```
            try: print(chain.chain[index])
```

```

        except: print("An issue occurred")

elif decide == "4":

    for b in chain.chain:

        print(b)

        print("-----")

elif decide == "5":

    if chain.validate(): print("Integrity validated.")

else:

    break

if __name__ == "__main__":

    manager()

```

Chapter 6: Input/ Output Screens

```
In [8]: runfile('C:/Users/Administrator/Desktop/mm.py', wdir='C:/Users/Administrator/Desktop')
```

Implementation of a Blockchain. Once recorded cannot be reversed.

Choice set:

- add message to the existing block (1)
- add existing block to the chain (2)
- show a block (index will be asked) (3)
- show the whole chain (4)
- validate the chain integrity (5)
- exit the program (6)

Fig 6.1: Choice Set

```
In [9]: runfile('C:/Users/Administrator/Desktop/mm.py', wdir='C:/Users/Administrator/Desktop')
```

Implementation of a Blockchain. Once recorded cannot be reversed.

Choice set:

- add message to the existing block (1)
- add existing block to the chain (2)
- show a block (index will be asked) (3)
- show the whole chain (4)
- validate the chain integrity (5)
- exit the program (6)

Your action: 1

Enter your data:Apple

Your action: 2

Your action: 3

Provide the index: 0

Block<hash: a2e532f3d1fd324f08f1a3abf4590c69236d2665fb90fcdb61e34dcf8267ffc5, prev_hash: None, messages: 1, time: 1608993615.852388>

Fig 6.2: Displaying a Block

```

Your action: 1
Enter your data:Apple

Your action: 1
Enter your data:Mango

Your action: 2

Your action: 1
Enter your data:Banana

Your action: 2

Your action: 3

Provide the index: 1
Block<hash: 1e2edaac374ece9742674ca6af87a25b8223a8388ea092812b99f8a9dc906ed1, prev_hash:
cacf75f79a408b1c4ed69f102ed4c9ee54a79e73a44cdc807d779904dbec2d97, messages: 1, time: 1608993857.3031979>

```

```

Your action: 4
Block<hash: cacf75f79a408b1c4ed69f102ed4c9ee54a79e73a44cdc807d779904dbec2d97, prev_hash: None, messages: 2, time:
1608993841.07827>
-----
Block<hash: 1e2edaac374ece9742674ca6af87a25b8223a8388ea092812b99f8a9dc906ed1, prev_hash:
cacf75f79a408b1c4ed69f102ed4c9ee54a79e73a44cdc807d779904dbec2d97, messages: 1, time: 1608993857.3031979>
-----

```

Fig 6.3 & 6.4: Displaying the blockchain with 2 blocks

Chapter 7: Conclusion

Out of this whole idea of implementing blockchain with python, the conclusion which we can draw is blockchain is a technology which holds in itself great potential to make remarkable changes in our lives. In a world where trust without using a third party is required, blockchain technology promises us this thing by being decentralized in nature. So in this whole project what we did is we implemented this whole concept in a very simple manner so that it can be understood by anyone and what we learnt is how this technology is structured at its very core.

9.1 Limitations

Although this project can be developed and advanced further, but at present one of the limitation of the project is that it could have been even more user interactive or user friendly provided if we would have incorporated features like that of Proof-of-work, distributed peer-to-peer network or mining of the block with the data.

For a person who has got no idea of how exactly does blockchain works, this project is helpful enough to make him/her understand how exactly it is structured at its very core. But the only limitation which we find over here is that it can be advanced even further and can be even more user interactive.

Apart from this project some of the limitations of blockchain are:

1. Mining uses excessive energy
2. Blockchain is indestructible
3. Scalability too is one of its weakness
4. Immutable (advantage and disadvantage at the same time)

9.2 Scope for Future Work

This project in itself holds a large future scope. It can be developed and advanced further. We can implement some of the advanced features like proof-of-work, consensus protocol, distributed peer-to-peer network or mining of the block with the data. The blockchain can be made even more user

intercative and user friendly. Blockchain as such has a fantastic future ahead in various fields like banking sector, healthcare, supply chain management , internet of things, cybersecurity etc. This field has also got a huge prospective of creating new job opportunities in this industry.

Bibliography

Bibliography

Blockchain. (2020, December 26th). Retrieved from Wikipedia: <https://en.wikipedia.org/wiki/Blockchain>

Blockchain. What is blockchain technology? How does it work? (2020, December 25th). Retrieved from builtin.com: <https://builtin.com/blockchain>

Jesse Yli-Huumo, D. K. (2016, October 3). *Where Is Current Research on Blockchain Technology?—A Systematic Review*. Retrieved from <https://www.researchgate.net/>:
https://www.researchgate.net/publication/308877750_Where_Is_Current_Research_on_Blockchain_Technology-A_Systematic_Review

Annexures



Plagiarism Report:

URKUND

Document Information

Analyzed document	Oshi Jain Minor Report.docx (D90627358)
Submitted	12/27/2020 4:48:00 AM
Submitted by	Siddhanta Kumar Singh
Submitter email	sksingh.set@modyuniversity.ac.in
Similarity	2%
Analysis address	sksingh.set.modyun@analysis.urkund.com

Sources included in the report

W	URL: https://httpdot.net/peers/Blockchains/blockchainIs-sourceText_txt-lf-utf8.txt Fetched: 11/4/2019 8:38:04 PM	 1
W	URL: https://en.wikipedia.org/wiki/Blockchain Fetched: 12/27/2020 4:49:00 AM	 1