JUNAID GIRKAR
60004190057
BE COMPS A2

BLOCKCHAIN TECHNOLOGIES

EXPERIMENT - 1

AIM: Implement and demonstrate Blockchain using Python

DEPLOYED LINK: **https://junaidgirkar.pythonanywhere.com/**

THEORY:
Blockchain is a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a business network.

Wherever there is an issue of trust, Blockchain can be implemented.

Steps in the working of Blockchain:

1. Transactions are stored in a place where they are verified.
2. When the number of transactions is > 500, they are combined together in one root hash value through a Merkle tree which is stored in a block.
3. Each block is connected to the ones before and after it rendering the blockchain tamper-evident

Components of a Blockchain network:-

1. Node
2. Ledger
3. Wallet
4. Nonce
5. Hash

1. **Node:**
   It is of two types – Full Node and Partial Node.
   - Full Node: It maintains a full copy of all the transactions. It has the capacity to validate, accept and reject transactions.
   - Partial Node: It is also called a Lightweight Node because it doesn't maintain the whole copy of the blockchain ledger. It maintains only the hash value of the transaction. The whole transaction is accessed using this hash value only. These nodes have low storage and low computational power.

2. **Ledger:**
   It is a digital database of information. Here, we have used the term 'digital' because the currency exchanged between different nodes is digital i.e cryptocurrency.

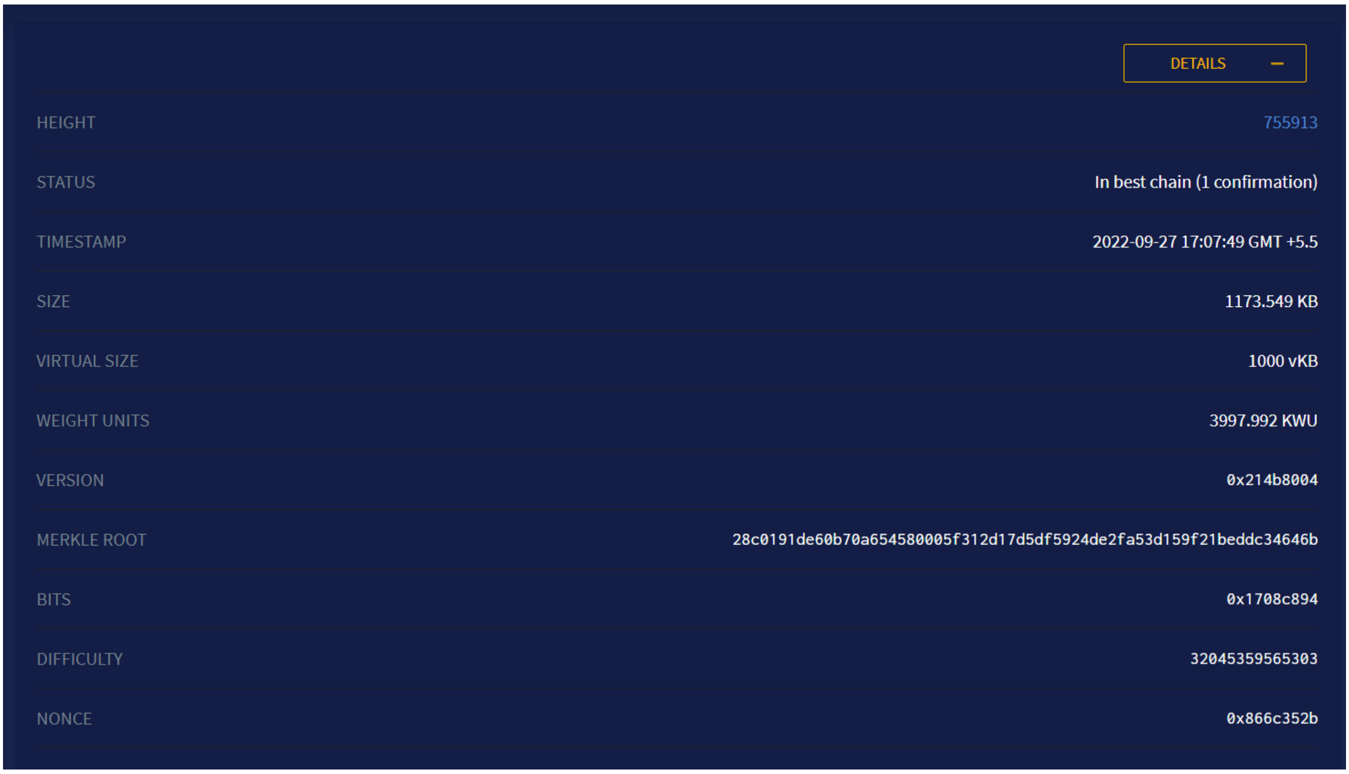
3. **Wallet:**
   It is a digital wallet that allows user to store their cryptocurrency. Every node in the blockchain network has a Wallet. Privacy of a wallet in a blockchain network is maintained using public and private key pairs. In a wallet, there is no need for currency conversion as the currency in the wallet is universally acceptable. Privacy of a wallet is maintained using public and private key pairs. Transactions are made secure as a private key is used both to send fund and to open the encrypted message.

4. **Nonce:**
   A nonce is an abbreviation for "number only used once," which is a number added to a hashed or encrypted block in a blockchain. It is the 32-bit number generated randomly only one time that assists to create a new block or validate a transaction. It is used to make the transaction more secure.

   It is hard to select the number which can be used as the nonce. It requires a vital amount of trial-and-error. First, a miner guesses a nonce. Then, it appends the guessed nonce to the hash of the current header. After that, it rehashes the value and compares this to the target hash. Now it checks whether the resulting hash value meets the requirements or not. If all the conditions are met, it means that the miner has created an answer and is granted the block.

5. **Hash:**
   The data is mapped to a fixed size using hashing. It plays a very important role in cryptography. In a blockchain network hash value of one transaction is the input of another transaction. Our example makes use of the SHA-256 Algorithm for hashing.

DETAILS OF A BLOCK:



HEIGHT	755913
STATUS	In best chain (1 confirmation)
TIMESTAMP	2022-09-27 17:07:49 GMT +5.5
SIZE	1173.549 KB
VIRTUAL SIZE	1000 vKB
WEIGHT UNITS	3997.992 KWU
VERSION	0x214b8004
MERKLE ROOT	28c0191de60b70a654580005f312d17d5df5924de2fa53d159f21beddc34646b
BITS	0x1708c894
DIFFICULTY	32045359565303
NONCE	0x866c352b

CODE: https://github.com/junaidgirkar/Blockchain-Django

Views.py

```
import random
from django.shortcuts import render
from .models import Block, JGChain
import hashlib, json
from django.views.decorators.csrf import csrf_exempt
from django.forms.models import model_to_dict

# Create your views here.
def home_view(request):

    # render function takes argument - request
    # and return HTML as response
    return render(request, "home.html")

def verify_blockchain():
    chain = Block.objects.all().order_by('id')
    previous_hash_data = chain[0].current_hash
    for block in chain[1:]:
        # if(block.id == 1):
        #     continue
        if(block.previous_hash != previous_hash_data):
            return False, block.id - 1
        else:
            previous_hash_data = block.current_hash
    return True, None


def get_chain(request):
    chain = Block.objects.all().order_by('-id')
    if(len(chain)>0):
        secure, tampered_block_id = verify_blockchain()
        return render(request, "get_chain.html", {'chain': chain, 'secure': secure, 'tampered_block_id': tampered_block_id})
    else:
        return render(request, "get_chain.html", {'chain': chain, 'secure' : True})


# generate a hash of an entire block
def hash(block):
    # assuming obj is your model instance
    json_data = model_to_dict(block)
    if 'current_hash' in json_data:
        del json_data['current_hash']
    encoded_block = json.dumps(json_data, sort_keys=True).encode()
    second_encoded_block = hashlib.sha256(encoded_block).hexdigest().encode()

    return hashlib.sha256(second_encoded_block).hexdigest()

def get_latest_block():
    return Block.objects.all().order_by('-id')[0]

@csrf_exempt
def mine_block(request):
    if(request.method == 'GET'):
        return render(request, "mine_block.html")

    # get the data we need to create a block
    if(request.method == 'POST'):

        # GENESIS BLOCK
        if(len(Block.objects.all()) == 0):
```

```
        previous_hash =
'0000000000000000000000000000000000000000000000000000000000000000'
        data = request.POST.get('data')
        # nonce = request.POST.get('nonce')
        difficulty = request.POST.get('difficulty')

        new_block = Block.objects.create(previous_hash=previous_hash, data=data, difficulty=difficulty)

        current_hash = hash(new_block)

        nonce = random.randint(0, 17223792749)
        while(current_hash[0:int(difficulty)] != '0'*int(difficulty)):
            new_block.nonce = nonce = random.randint(0, 17223792749)
            new_block.save()
            current_hash = hash(new_block)
            print(current_hash)

        new_block.current_hash = current_hash
        new_block.nonce = nonce
        new_block.save()
        return render(request, "mine_block.html", {'MinedBlock': new_block})

    else:
        previous_block = get_latest_block()
        previous_hash = previous_block.current_hash
        data = request.POST.get('data')
        # nonce = request.POST.get('nonce')
        difficulty = request.POST.get('difficulty')

        new_block = Block.objects.create(previous_hash=previous_hash, data=data, difficulty=difficulty)
        current_hash = hash(new_block)

        nonce = random.randint(0, 17223792749)

        while(current_hash[0:int(difficulty)] != '0'*int(difficulty)):
            new_block.nonce = nonce = random.randint(0, 17223792749)
            new_block.save()
            current_hash = hash(new_block)
            print(current_hash)

        new_block.current_hash = current_hash
        new_block.nonce = nonce
        new_block.save()
```

```
            return render(request, "mine_block.html", {'MinedBlock': new_block})


def block_detail_view(request, id):
    block = Block.objects.get(id=id)
    return render(request, "block_detail.html", {'detail_block': block})


def attack_a_block(request):
    if(request.method == 'GET'):
        return render(request, "attack_a_block.html")

    elif(request.method == 'POST'):
        block_id = request.POST.get('block_id')
        block = Block.objects.get(id=block_id)
        block.data = request.POST.get('MaliciousData')
        block.save()
        current_hash = hash(block)
        block.current_hash = current_hash
        block.save()

        return render(request, "attack_a_block.html", {'AttackedBlock': block})

@csrf_exempt
def delete_all_blocks(request):
    Block.objects.all().delete()
    return render(request, "get_chain.html")
```

Models.py

```
from django.db import models

# Create your models here.
class Block(models.Model):
    id = models.AutoField(primary_key=True)
    current_hash = models.CharField(max_length=64, null=False, blank=False)
    previous_hash = models.CharField(max_length=64, null=False, blank=False)
    timestamp = models.DateTimeField(auto_now_add=True)
    data = models.TextField(blank=True, null=True)
    nonce = models.IntegerField(default=0)
    difficulty = models.IntegerField(default=0)
```

Urls.py

```
from .views import mine_block, get_chain, block_detail_view, attack_a_block, delete_all_blocks
from django.urls import path

app_name = "JGChain"
urlpatterns = [
    path('', get_chain, name="full_chain"),
    path('mine_block/', mine_block, name="mine_block"),
    path('detail/<int:id>/', block_detail_view, name="block_detail_view"),
    path('attack/', attack_a_block, name="attack_block"),
    path('delete_all_blocks/', delete_all_blocks, name="delete_all_blocks"),
]
```

Base.html

```
<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]>      <html class="no-js"> <!--<![endif]-->
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width; initial-scale=1.0; maximum-scale=1.0;" />
        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js"
integrity="sha384-B0UglyR+jN6CkvvICOB2joaf5I4l3gm9GU6Hc1og6Ls7i6U/mkkaduKaBhlAXv9k"
crossorigin="anonymous"></script>

        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU"
crossorigin="anonymous">
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js"
integrity="sha384-/bQdsTh/da6pkI1MST/rWKFNjaCP5gBSY4sEBT38Q/9RBh9AH40zEOg7Hlq2THRZ"
```

```
crossorigin="anonymous"></script>

  </head>
  <body>
    <nav class="navbar sticky-top navbar-expand navbar-dark bg-dark px-3">
      <a class="navbar-brand" href="#">JGChain</a>
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-bs-target="#navbarText" aria-controls="navbarText" aria-expanded="false" aria-label="Toggle navigation">
       <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse mr-auto" id="navbarText">
       <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
         <a class="nav-link" href="{% url 'JGChain:full_chain' %}">Full Chain </a>
        </li>
        <li class="nav-item">
         <a class="nav-link" href={% url 'JGChain:mine_block' %}>Mine a New Block</a>
        </li>

        <li class="nav-item">
         <a class="nav-link" href={% url 'JGChain:attack_block' %}>Attack a Block</a>
        </li>


      </div>
       </ul>
       <div class="ml-auto">
       <span class="navbar-text">
        Created by Junaid Girkar
       </span>

      </div>
     </nav>

     <br>
     {% block content %}{% endblock content %}
  </body>
</html>
```

get_chain.html

```
{% extends "base.html" %}
{% block content %}

<div class="container">

    {% if chain %}
        {% if secure %}
        <div class="alert alert-success">
        <strong>Success!</strong> THE CHAIN HAS NOT BEEN TAMPERED WITH.
        </div>

        {% else %}
        <div class="alert alert-danger">
            <strong>Danger!</strong> THE CHAIN HAS BEEN TAMPERED AT BLOCK ID {{ tampered_block_id }}.
          </div>
        {% endif %}

        <form action={% url 'JGChain:delete_all_blocks' %}  method="POST">
        <button type="submit" class="btn btn-warning">Delete All Blocks</button>
        </form>
        <br><br>
    {% else %}

    <div class="alert alert-primary" role="alert">
        There are 0 blocks currently mined!
      </div>

    {% endif %}
        {% for item in chain %}

        {% if item.id != tampered_block_id %}
            <div class="row alert container">
                <div class="col">
                <a href = {% url 'JGChain:block_detail_view' id=item.id %}> <h4>Block ID : {{ item.id}} </h4> </a>
                <p>Previous Hash : {{ item.previous_hash}} </p>
                <p>Current Hash : {{ item.current_hash}} </p>

                <p>Timestamp : {{ item.timestamp}} </p>
                <p>Nonce : {{ item.nonce}} </p>
                <p>Difficulty : {{ item.difficulty}} </p>

                <p>Block Data : {{ item.data}} </p>
                </div>
            </div>
            <hr>
        {% else %}
        <div class="row alert text-danger container">
            <a href = {% url 'JGChain:block_detail_view' id=item.id %}> <h4>Block ID : {{ item.id}} </h4> </a>
            <p>Previous Hash : {{ item.previous_hash}} </p>
            <p>Current Hash : {{ item.current_hash}} </p>

            <p>Timestamp : {{ item.timestamp}} </p>
            <p>Nonce : {{ item.nonce}} </p>
            <p>Difficulty : {{ item.difficulty}} </p>

            <p>Block Data : {{ item.data}} </p>
        </div>
        <hr>
        {% endif %}

        {% endfor %}
</div>
{% endblock content %}
```

block_detail.html

```
{% extends "base.html" %}

{% block content %}

  <div class="container">
    <h4>Block ID : {{ detail_block.id }} </h4>
    <p>Previous Hash : {{ detail_block.previous_hash }} </p>
    <p>Current Hash : {{ detail_block.current_hash }} </p>
    <p>Timestamp : {{ detail_block.timestamp }} </p>
    <p>Nonce : {{ detail_block.nonce }} </p>
    <p>Difficulty : {{ detail_block.difficulty}} </p>
    <p>Block Data : {{ detail_block.data }} </p>
  </div>

{% endblock content %}
```

mine_block.html

```
{% extends "base.html" %}
{% block content %}

  <div style="margin: auto; width: 50%; border: 3px solid green; padding: 10px;">
  <form action={% url 'JGChain:mine_block' %}  method="POST">
    {% csrf_token %}

    <div class="form-group row col-sm-12">

      <label for="difficulty" class="col-sm-2 col-form-label">Difficulty</label>
      <div class="col-sm-10">
       <input type="number" class="form-control" id="difficulty" name="difficulty" value=0 min="0" max="3">
      </div>

      <br><br><br>
      <label for="data" class="col-sm-2 col-form-label">Transactions</label>
      <div class="col-sm-10">
        <textarea class="form-control" id="data" name="data" rows="3"></textarea>
      </div>

      <br>
      <br><br><br>
      <div class="container">
        <div class="row">
         <div class="col text-center">
          <input type="submit" class="btn btn-success" value="Submit">
         </div>
        </div>
       </div>
    </div>
  </div>
  </form>
  </div>

  <hr>

  {% if MinedBlock %}

  <div class="block container">
   <h2> MINED BLOCK </h2><br><br>

   <a href = {% url 'JGChain:block_detail_view' id=MinedBlock.id %}> <h4>Block ID : {{ MinedBlock.id}} </h4> </a>
```

```
    <p>Current Hash : {{ MinedBlock.current_hash}} </p>
    <p>Previous Hash : {{ MinedBlock.previous_hash}} </p>
    <p>Timestamp : {{ MinedBlock.timestamp}} </p>
    <p>Nonce : {{ MinedBlock.nonce}} </p>
    <p>Difficulty : {{ MinedBlock.difficulty}} </p>
    <p>Block Data : {{ MinedBlock.data}} </p>
  </div><br>
  {% endif %}

  {% endblock content%}
```

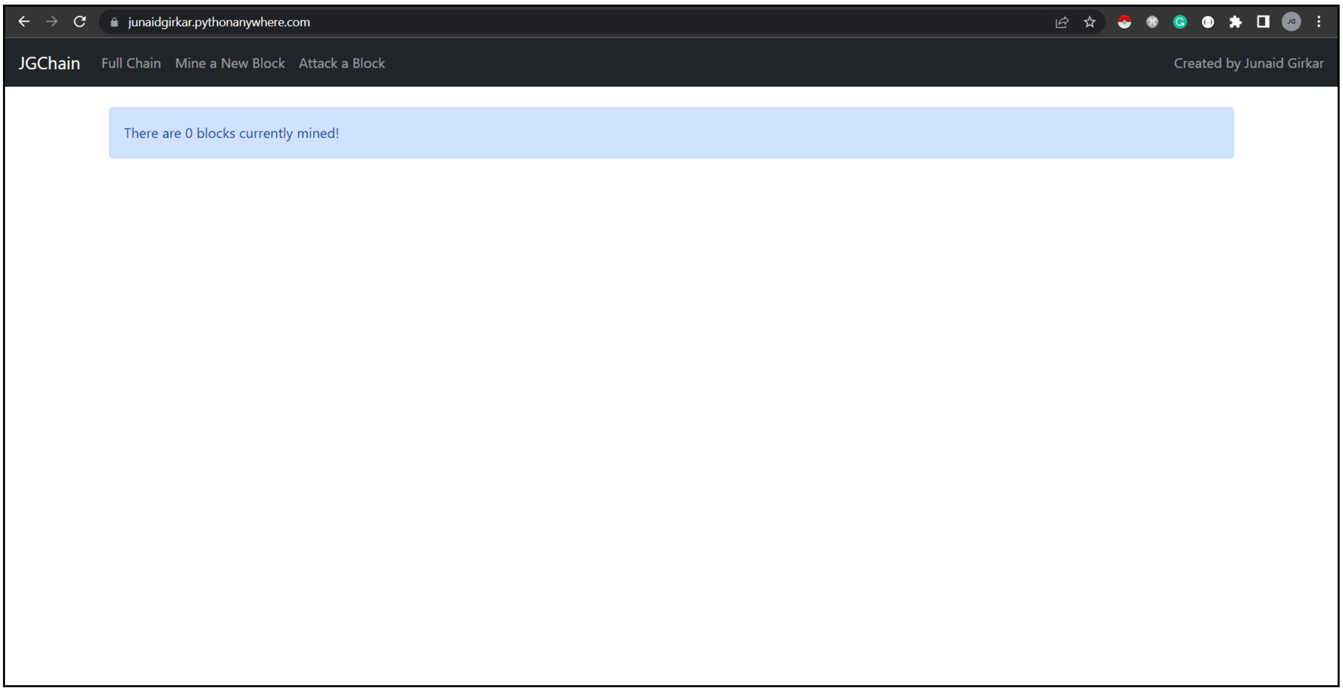
attack_a_block.html

```
{% extends "base.html" %}

{% block content %}

<div style="margin: auto; width: 50%; border: 3px solid red; padding: 10px;">
  <form action={% url 'JGChain:attack_block'%}  method="POST">
    {% csrf_token %}

    <div class="form-group row col-sm-12">
      <label for="block_id" class="col-sm-2 col-form-label">Block ID</label>
      <div class="col-sm-10">
       <input type="number" class="form-control" id="block_id" name="block_id" value=0>
      </div>


      <br><br><br>
      <label for="data" class="col-sm-2 col-form-label">Malicious Data</label>
      <div class="col-sm-10">
        <textarea class="form-control" id="MaliciousData" name="MaliciousData"
rows="3"></textarea>
      </div>

      <br>
      <br>
      <br>
      <br>

      <div class="container">
        <div class="row">
```

```
         <div class="col text-center">
          <input type="submit" class="btn btn-danger" value="Attack">
         </div>
        </div>
      </div>
    </div>
  </form>
</div>

<hr>


{% if AttackedBlock %}

<div class="block container">
 <h2> ATTACKED BLOCK </h2><br><br>

 <a href = {% url 'JGChain:block_detail_view' id=AttackedBlock.id %}> <h4>Block ID : {{
AttackedBlock.id}} </h4> </a>
  <p>Current Hash : {{ AttackedBlock.current_hash}} </p>
  <p>Previous Hash : {{ AttackedBlock.previous_hash}} </p>
  <p>Timestamp : {{ AttackedBlock.timestamp}} </p>
  <p>Nonce : {{ AttackedBlock.nonce}} </p>
  <p>Difficulty : {{ AttackedBlock.difficulty}} </p>

  <p>Block Data : {{ AttackedBlock.data}} </p>
</div><br>
{% endif %}

{% endblock content %}
```

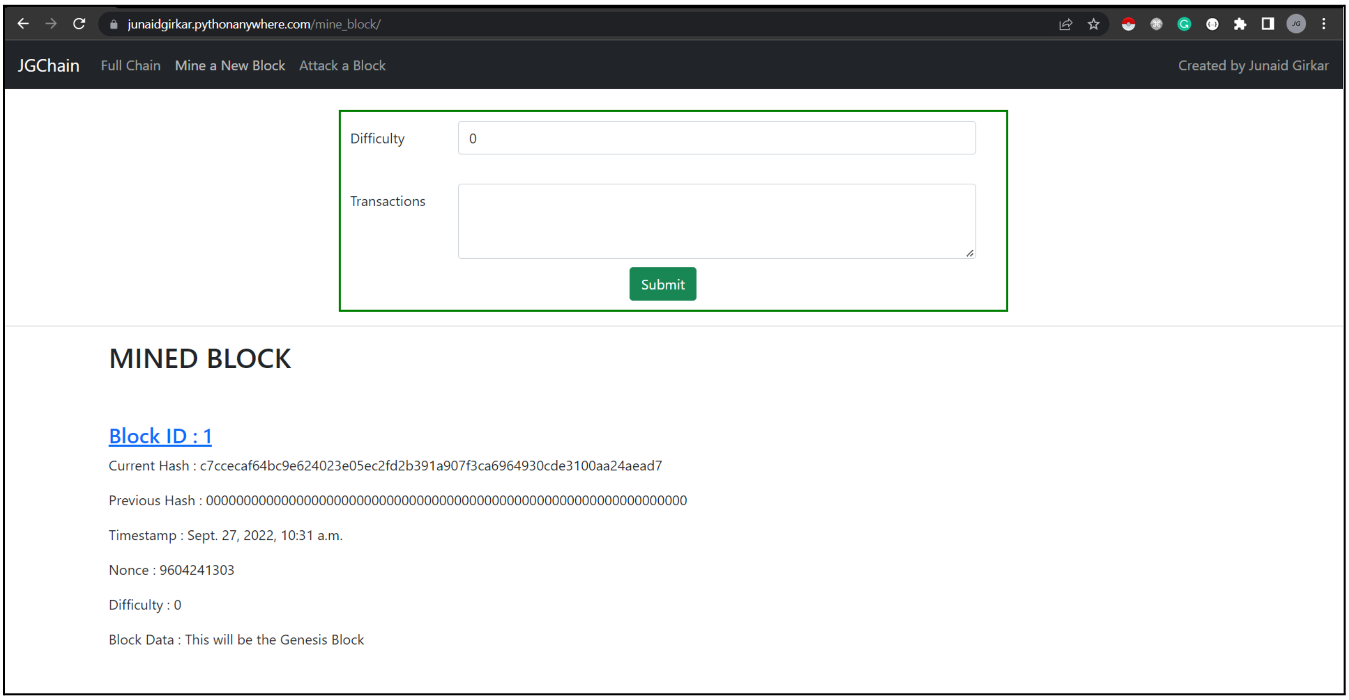
OUTPUT:

Initial:



There are 0 blocks currently mined!

Mine a block:



Difficulty 0

Transactions

MINED BLOCK

Block ID : 1

Current Hash : c7ccecaf64bc9e624023e05ec2fd2b391a907f3ca6964930cde3100aa24aead7

Previous Hash : 0000000000000000000000000000000000000000000000000000000000000000

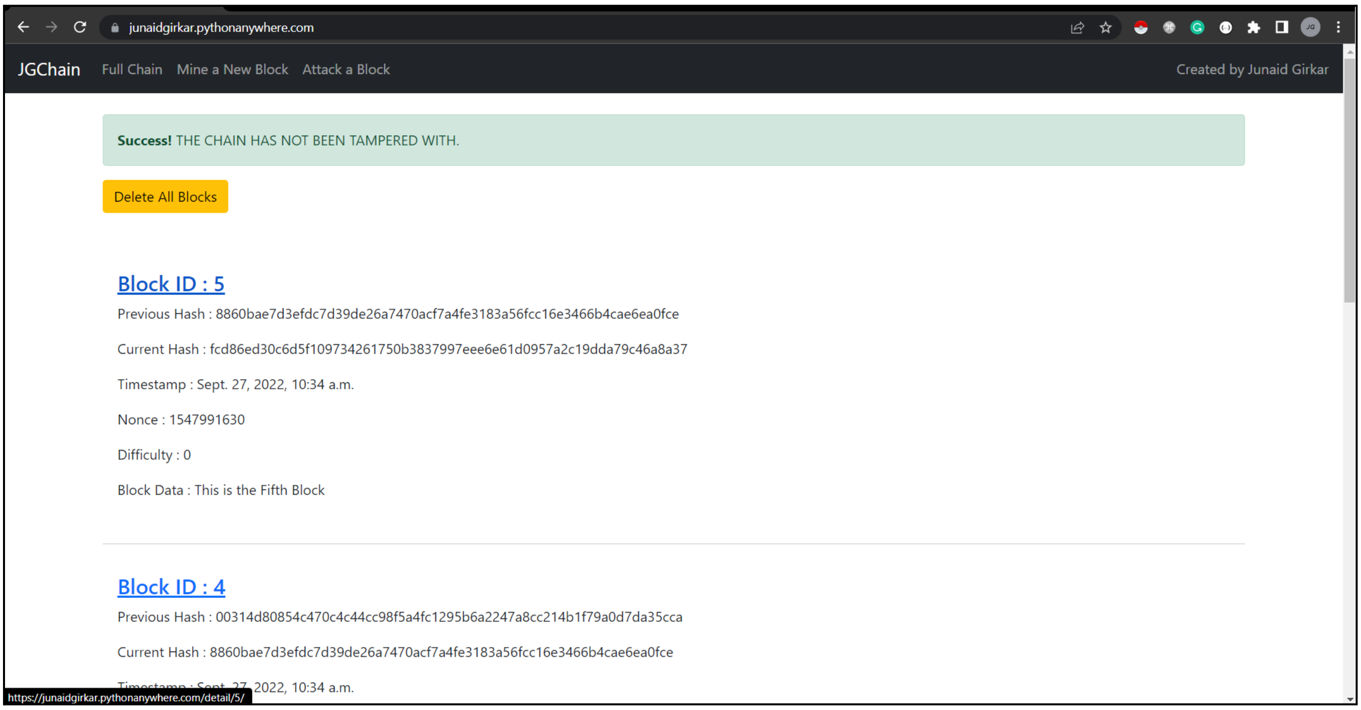
Timestamp : Sept. 27, 2022, 10:31 a.m.

Nonce : 9604241303

Difficulty : 0

Block Data : This will be the Genesis Block

After Mining 5 Blocks with different difficulties:



Success! THE CHAIN HAS NOT BEEN TAMPERED WITH.

Block ID : 5

Previous Hash : 8860bae7d3efdc7d39de26a7470acf7a4fe3183a56fcc16e3466b4cae6ea0fce

Current Hash : fcd86ed30c6d5f109734261750b3837997eee6e61d0957a2c19dda79c46a8a37

Timestamp : Sept. 27, 2022, 10:34 a.m.

Nonce : 1547991630

Difficulty : 0

Block Data : This is the Fifth Block

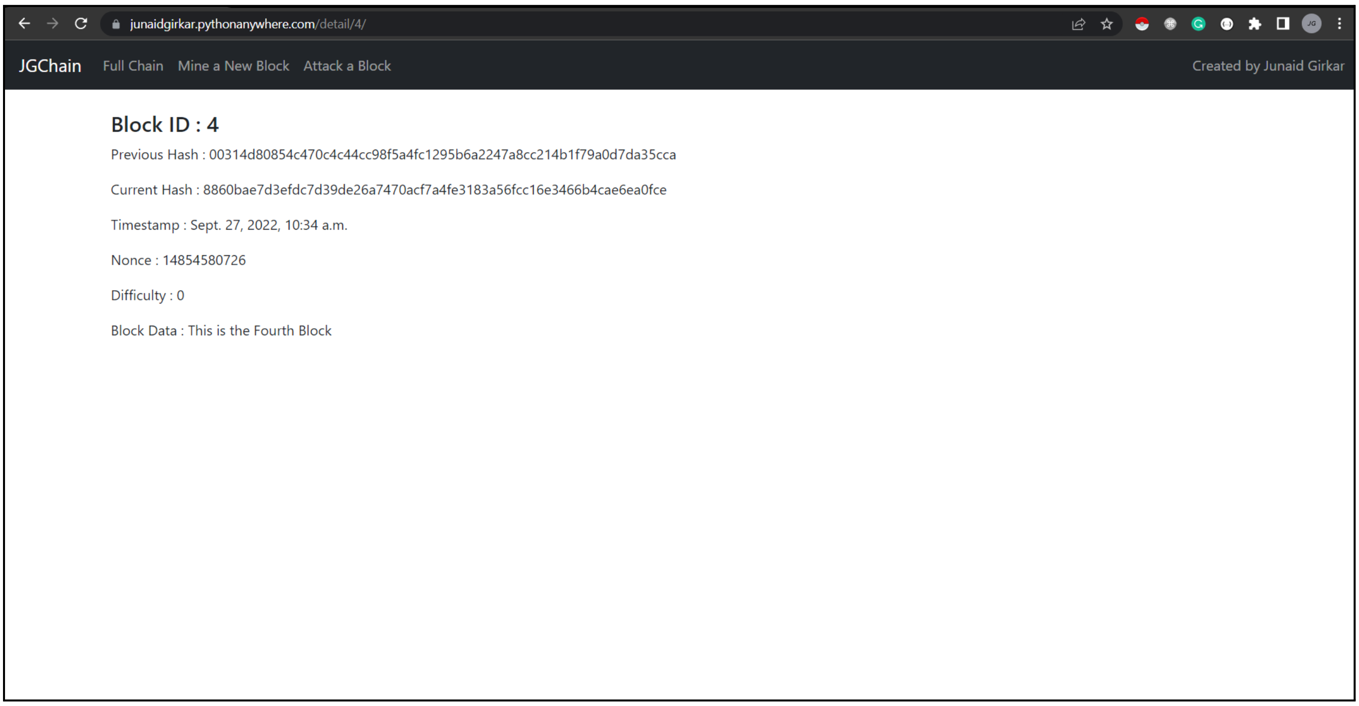
Block ID : 4

Previous Hash : 00314d80854c470c4c44cc98f5a4fc1295b6a2247a8cc214b1f79a0d7da35cca

Current Hash : 8860bae7d3efdc7d39de26a7470acf7a4fe3183a56fcc16e3466b4cae6ea0fce

Timestamp : Sept. 27, 2022, 10:34 a.m.

Detail View of each Block:



Block ID : 4

Previous Hash : 00314d80854c470c4c44cc98f5a4fc1295b6a2247a8cc214b1f79a0d7da35cca

Current Hash : 8860bae7d3efdc7d39de26a7470acf7a4fe3183a56fcc16e3466b4cae6ea0fce

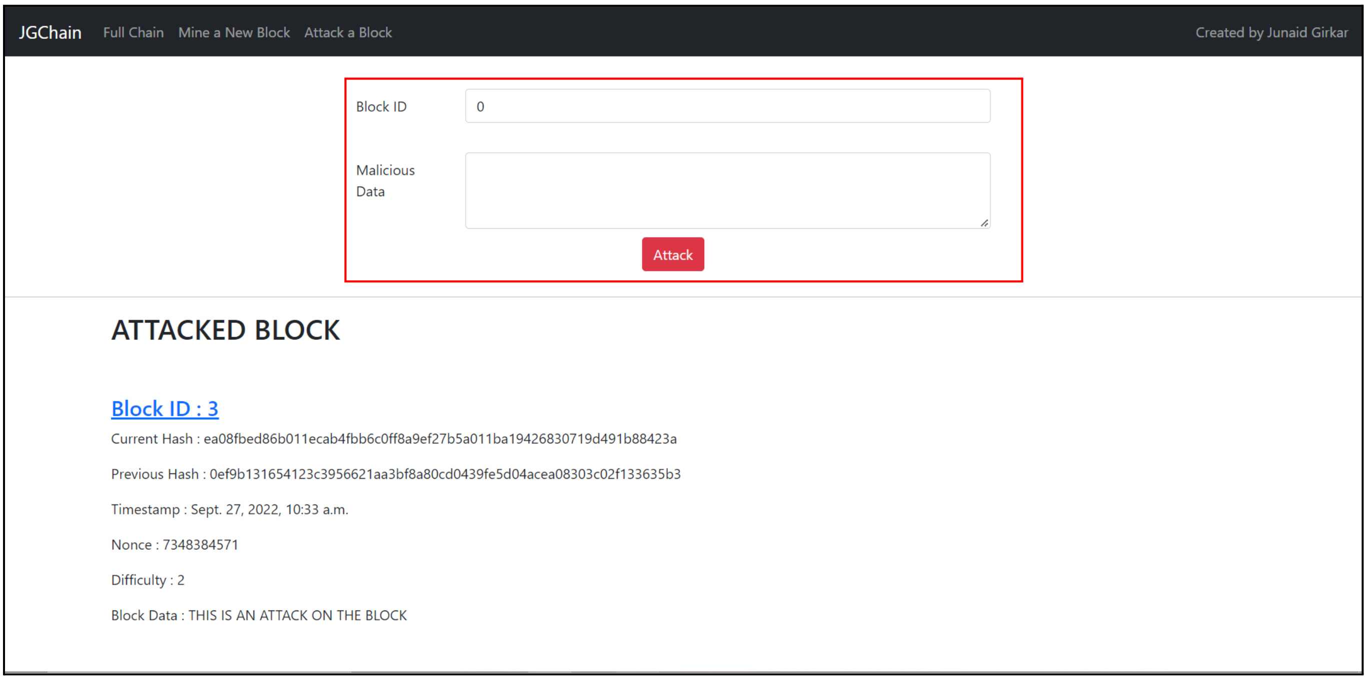
Timestamp : Sept. 27, 2022, 10:34 a.m.

Nonce : 14854580726

Difficulty : 0

Block Data : This is the Fourth Block

Attacking a specific block:



Block ID: 0

Malicious Data:

ATTACKED BLOCK

Block ID : 3

Current Hash : ea08fbed86b011ecab4fbb6c0ff8a9ef27b5a011ba19426830719d491b88423a

Previous Hash : 0ef9b131654123c3956621aa3bf8a80cd0439fe5d04acea08303c02f133635b3

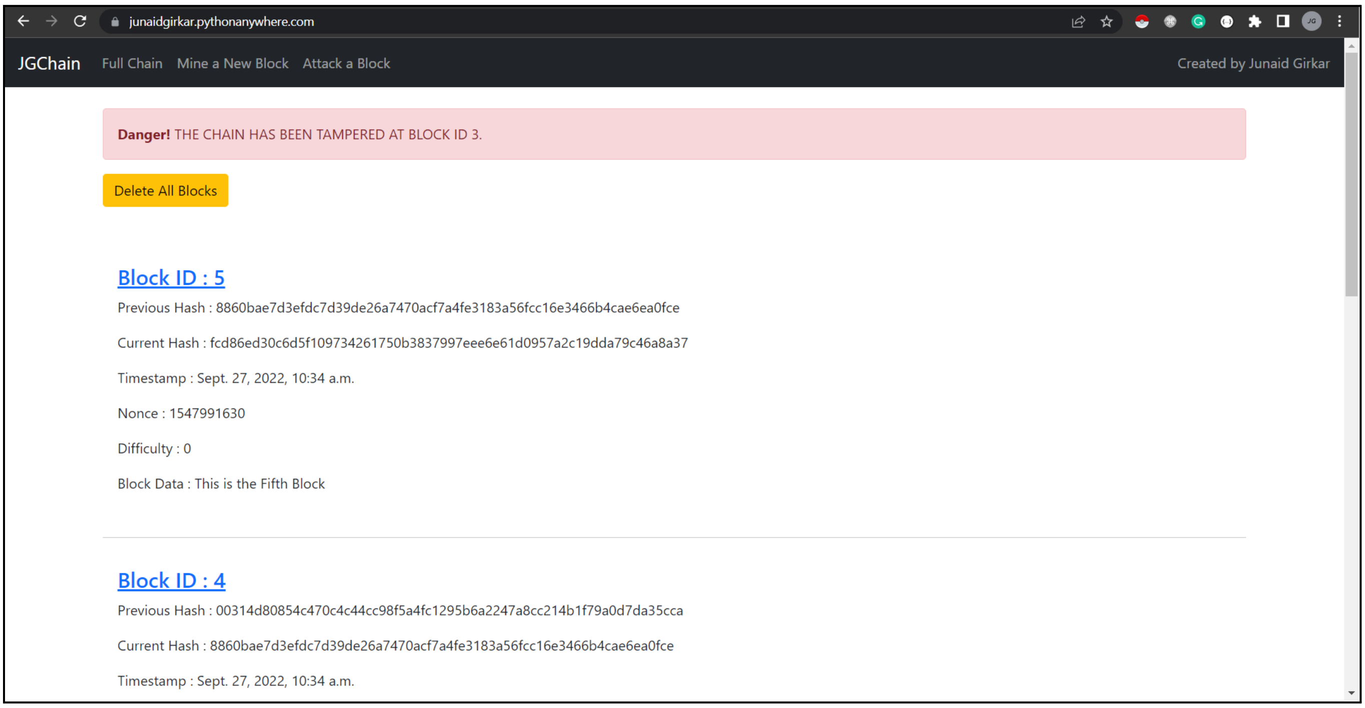
Timestamp : Sept. 27, 2022, 10:33 a.m.

Nonce : 7348384571

Difficulty : 2

Block Data : THIS IS AN ATTACK ON THE BLOCK

After attacking a block:



Danger! THE CHAIN HAS BEEN TAMPERED AT BLOCK ID 3.

Block ID : 5

Previous Hash : 8860bae7d3efdc7d39de26a7470acf7a4fe3183a56fcc16e3466b4cae6ea0fce

Current Hash : fcd86ed30c6d5f109734261750b3837997eee6e61d0957a2c19dda79c46a8a37

Timestamp : Sept. 27, 2022, 10:34 a.m.

Nonce : 1547991630

Difficulty : 0

Block Data : This is the Fifth Block

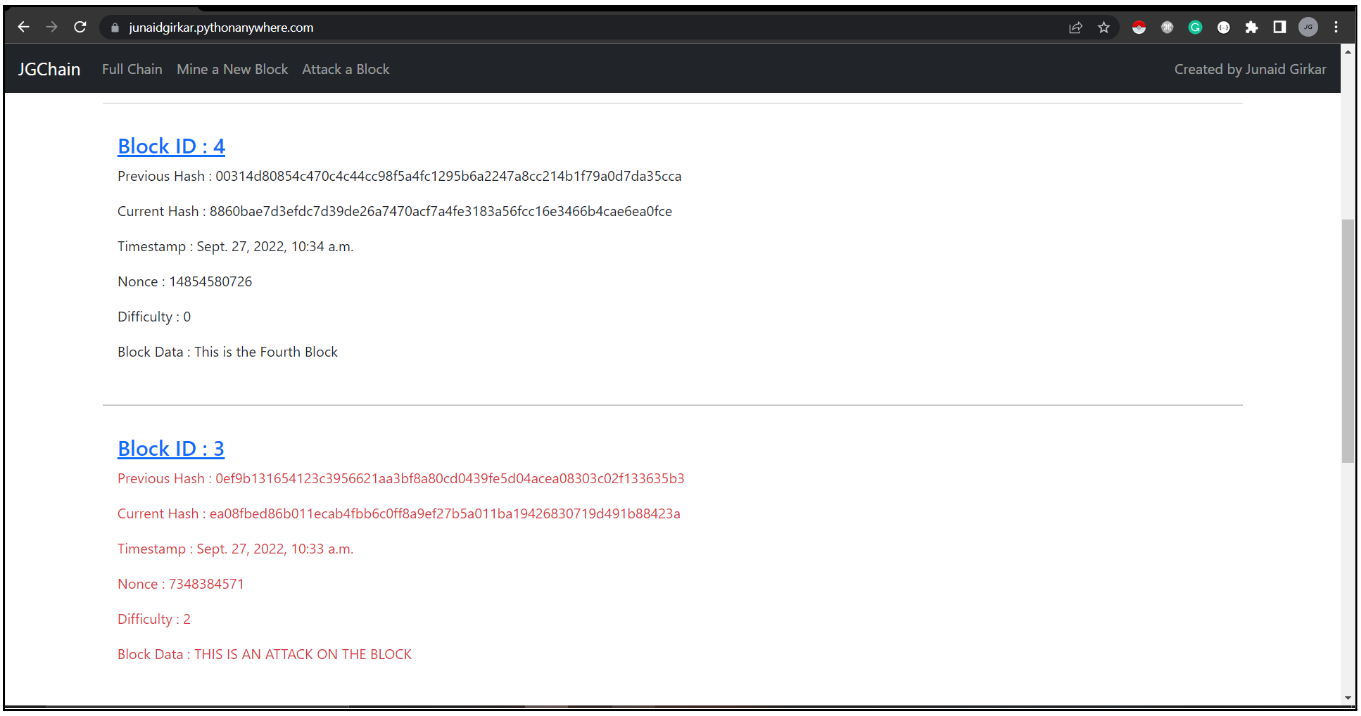
Block ID : 4

Previous Hash : 00314d80854c470c4c44cc98f5a4fc1295b6a2247a8cc214b1f79a0d7da35cca

Current Hash : 8860bae7d3efdc7d39de26a7470acf7a4fe3183a56fcc16e3466b4cae6ea0fce

Timestamp : Sept. 27, 2022, 10:34 a.m.

The previous hash of Block 4 not matching with the current hash of Block 3:



Block ID : 4

Previous Hash : 00314d80854c470c4c44cc98f5a4fc1295b6a2247a8cc214b1f79a0d7da35cca

Current Hash : 8860bae7d3efdc7d39de26a7470acf7a4fe3183a56fcc16e3466b4cae6ea0fce

Timestamp : Sept. 27, 2022, 10:34 a.m.

Nonce : 14854580726

Difficulty : 0

Block Data : This is the Fourth Block

Block ID : 3

Previous Hash : 0ef9b131654123c3956621aa3bf8a80cd0439fe5d04acea08303c02f133635b3

Current Hash : ea08fbed86b011ecab4fbb6c0ff8a9ef27b5a011ba19426830719d491b88423a

Timestamp : Sept. 27, 2022, 10:33 a.m.

Nonce : 7348384571

Difficulty : 2

Block Data : THIS IS AN ATTACK ON THE BLOCK

CONCLUSION:

In this practical, we learnt about blockchain, its types, features, benefits and its elements. We then implemented the core concept of blockchain in python using Django as the framework for a web interface.