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
6. **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.

1. **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.

1. **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.

1. **Nonce:**

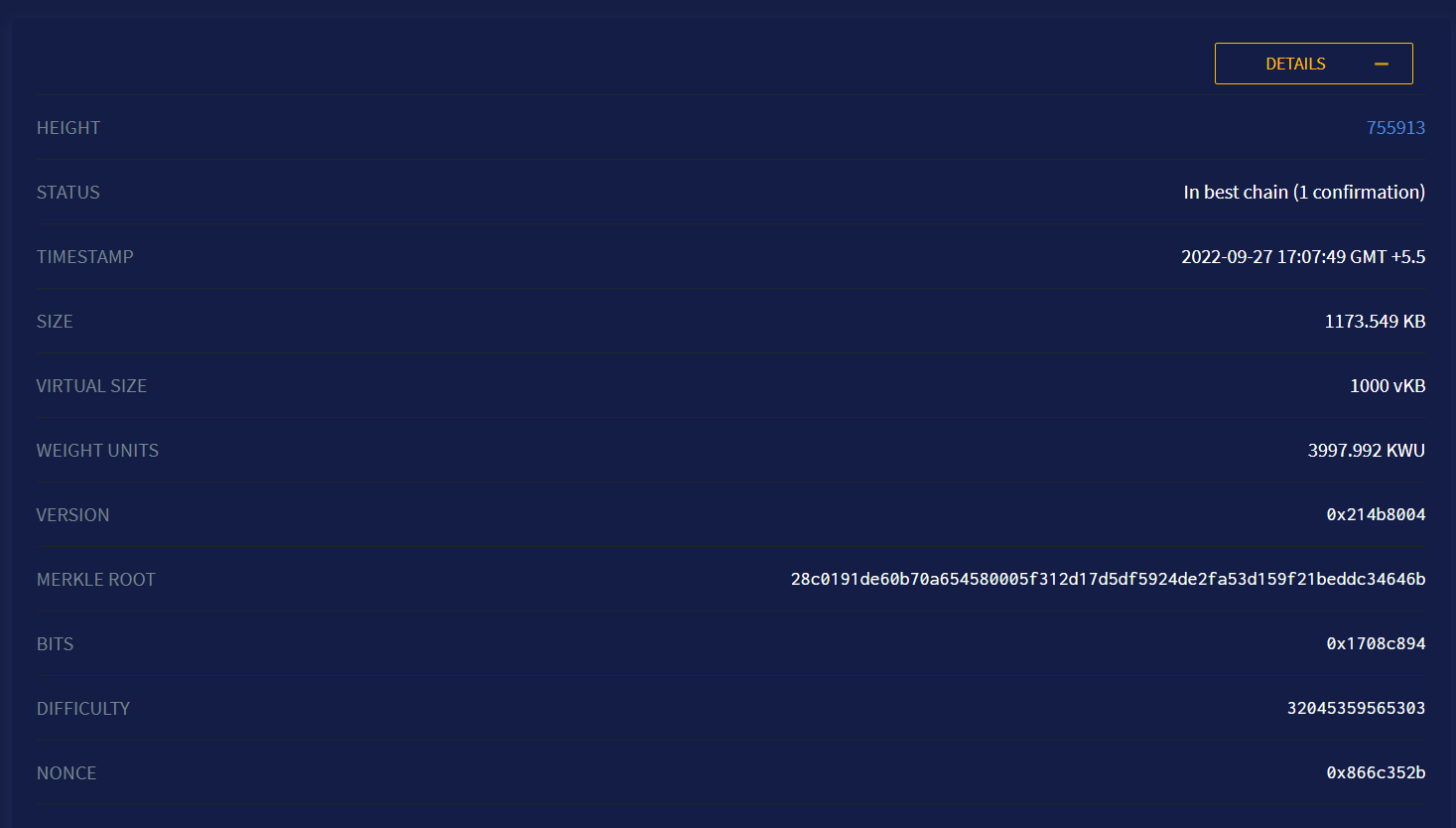
A nonce is an abbreviation for “number only used once,” which is a number added to a hashed or encrypted block in a blockchain. It is the 32-bit number generated randomly only one time that assists to create a new block or validate a transaction. It is used to make the transaction more secure.

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.

1. **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:



### 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

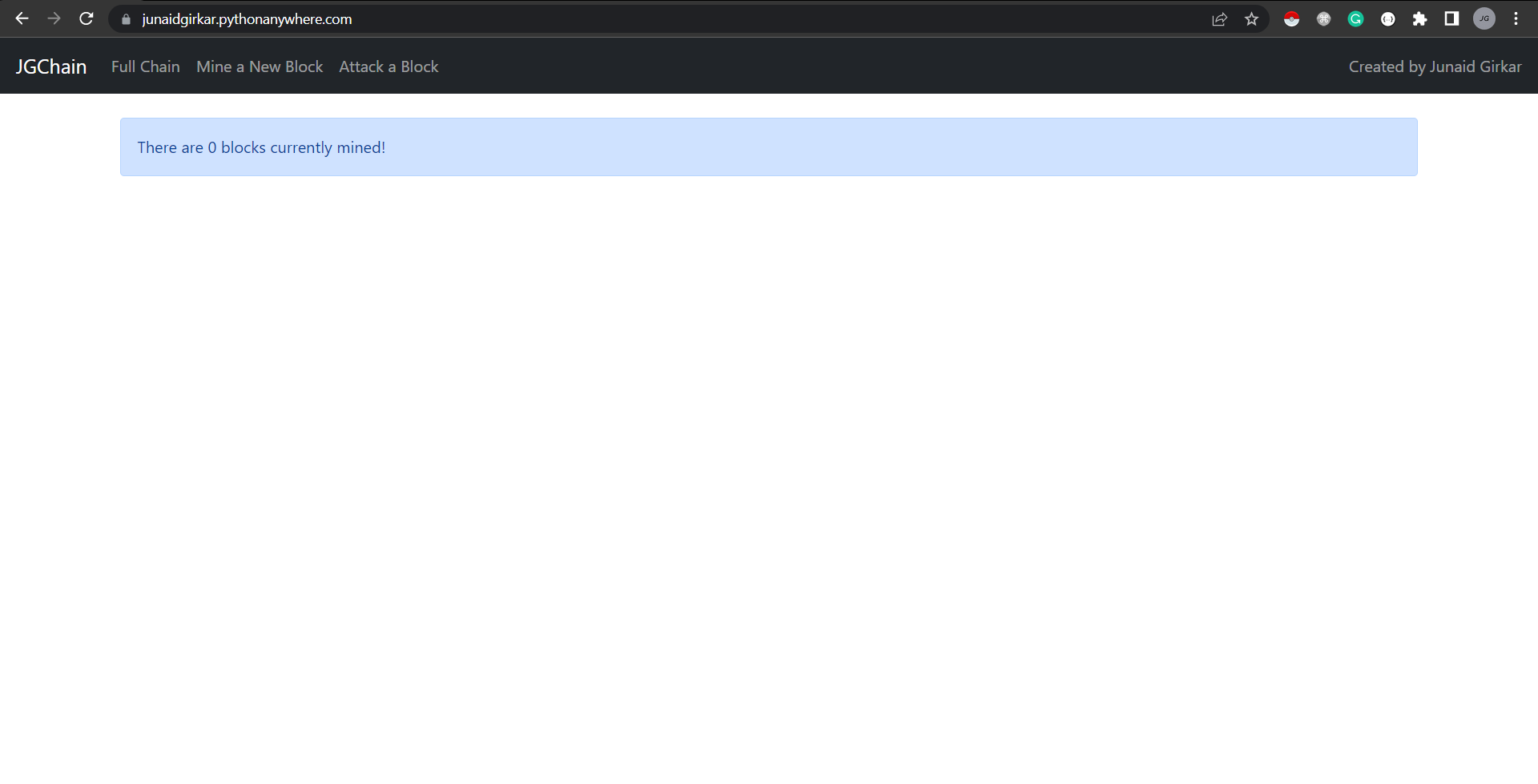
| {% extends "base.html" %}  {% block content %}   <div style="margin: auto; width: 50%; border: 3px solid green; padding: 10px;">  <form action={% url 'JGChain:mine\_block' %} method="POST">  {% csrf\_token %}   <div class="form-group row col-sm-12">   <label for="difficulty" class="col-sm-2 col-form-label">Difficulty</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

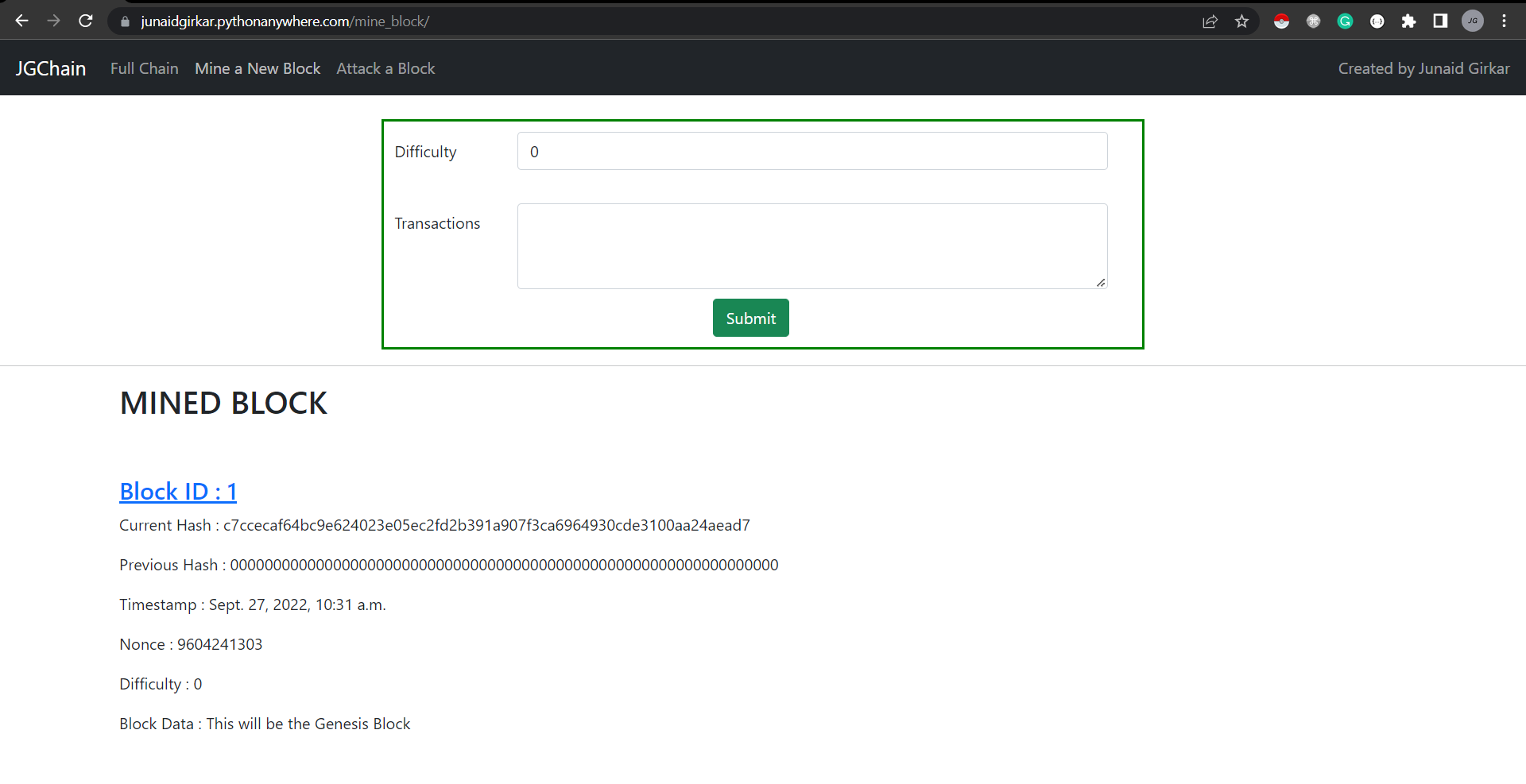
| {% extends "base.html" %}  {% block content %}  <div style="margin: auto; width: 50%; border: 3px solid red; padding: 10px;">  <form action={% url 'JGChain:attack\_block'%} method="POST">  {% csrf\_token %}   <div class="form-group row col-sm-12">  <label for="block\_id" class="col-sm-2 col-form-label">Block 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:

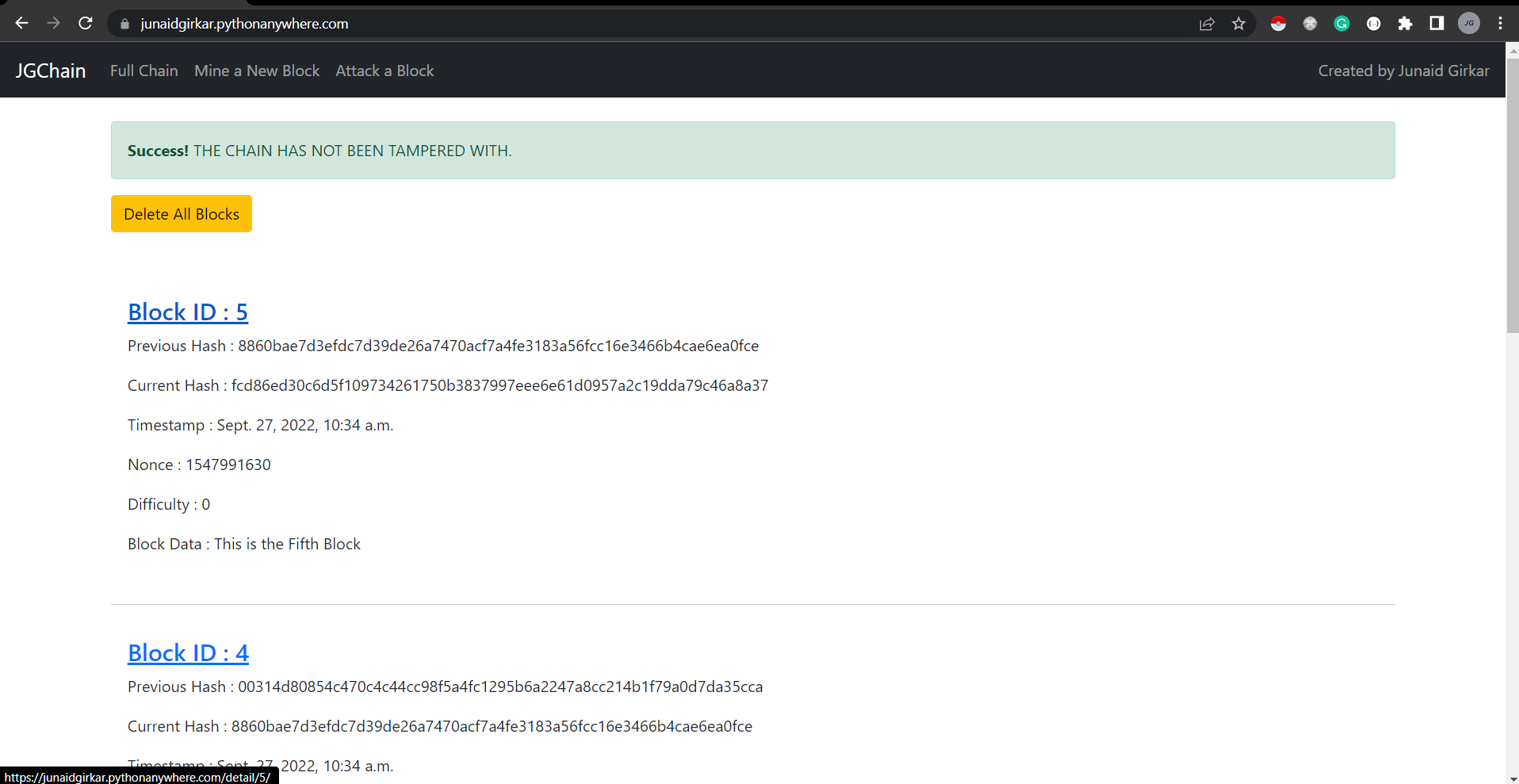
#### Initial:



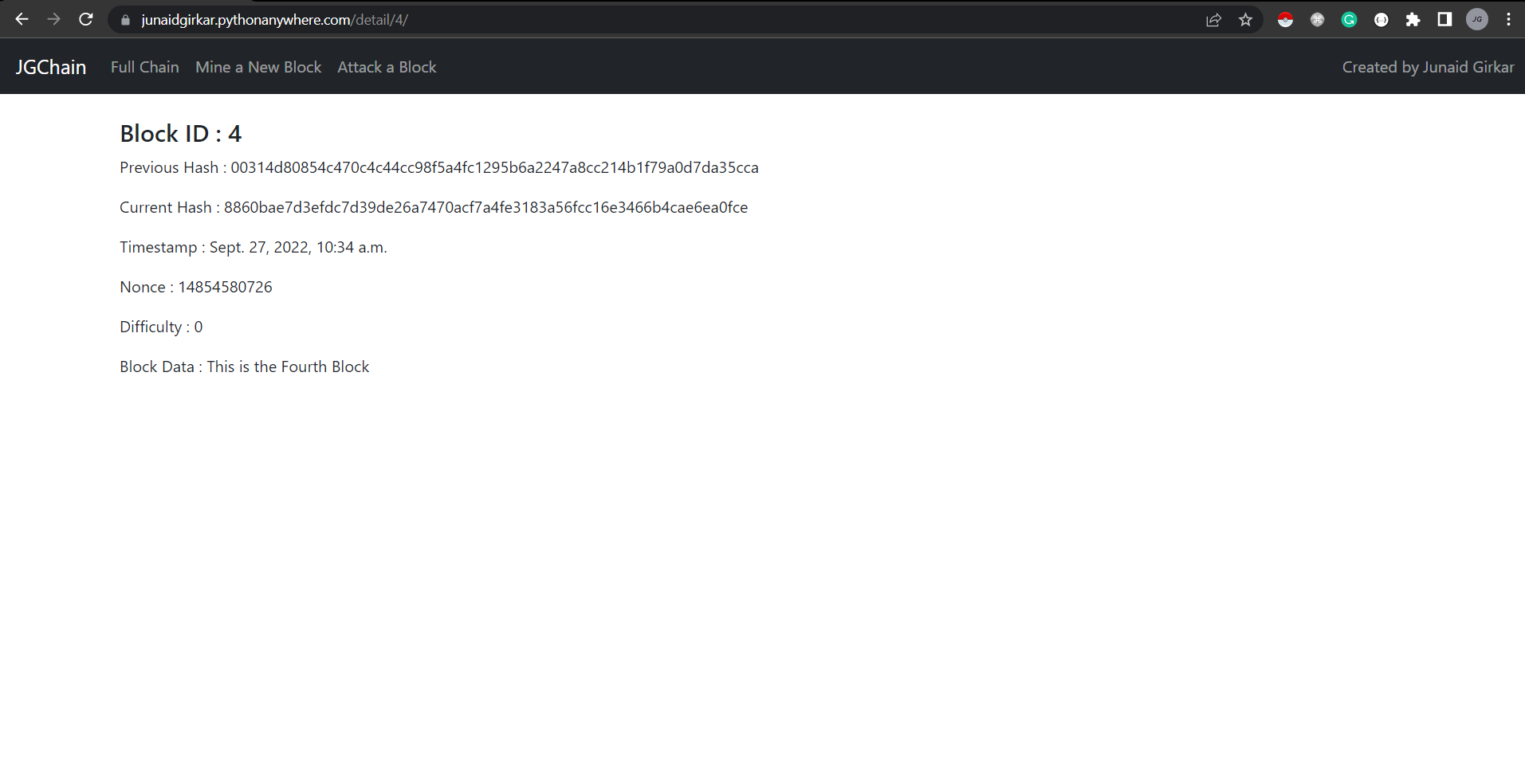
#### Mine a block:



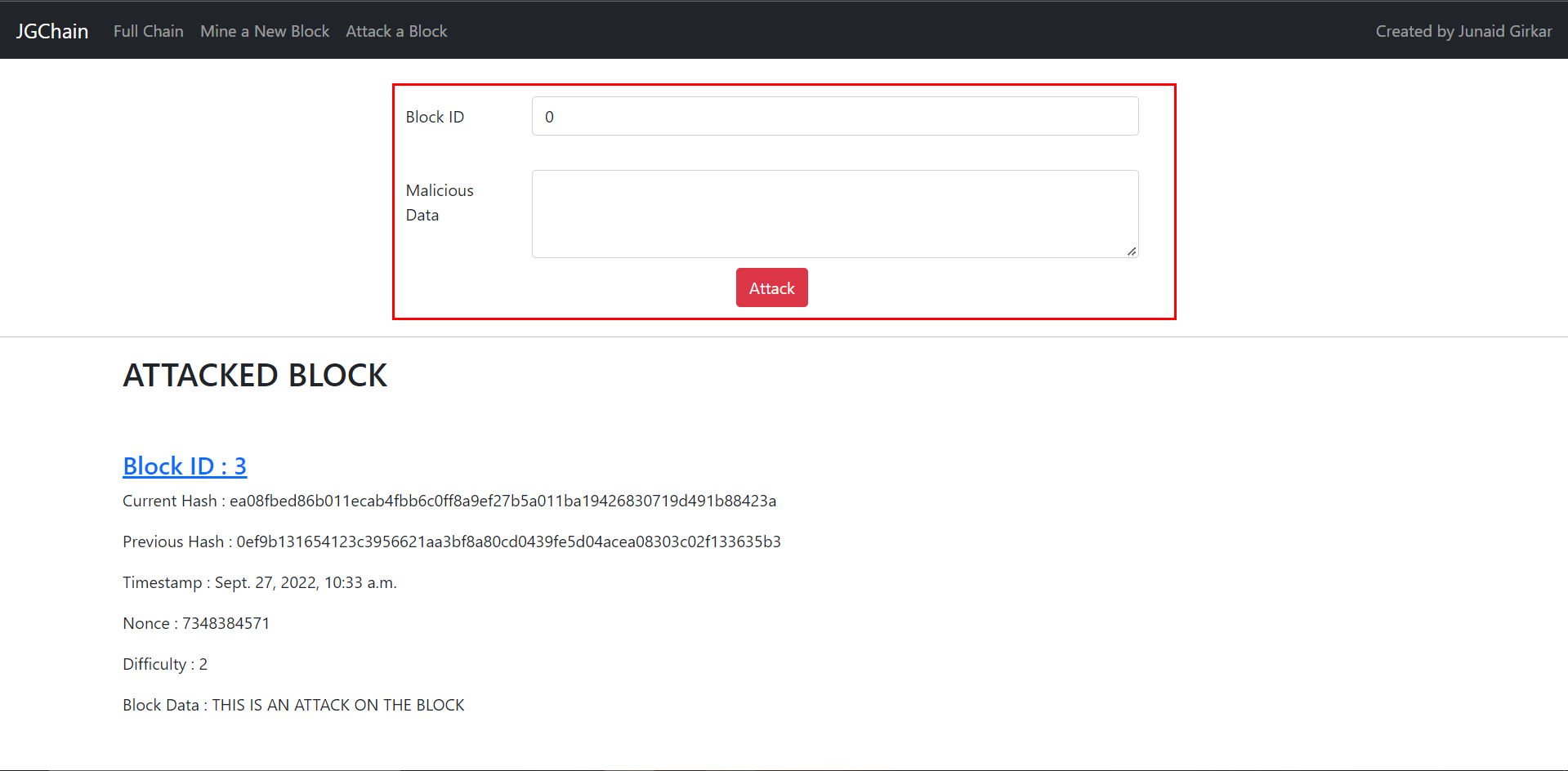
#### After Mining 5 Blocks with different difficulties:



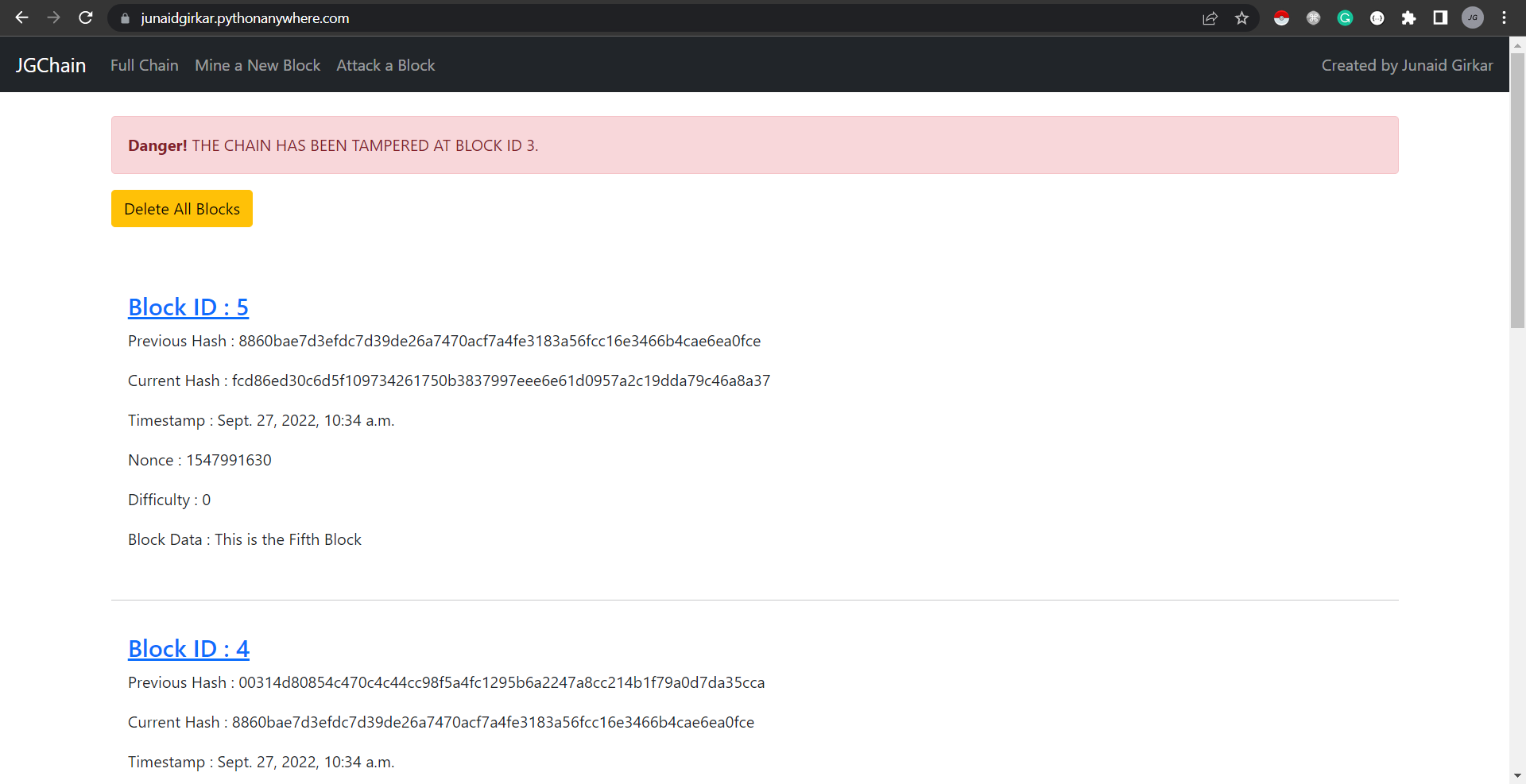
#### Detail View of each Block:



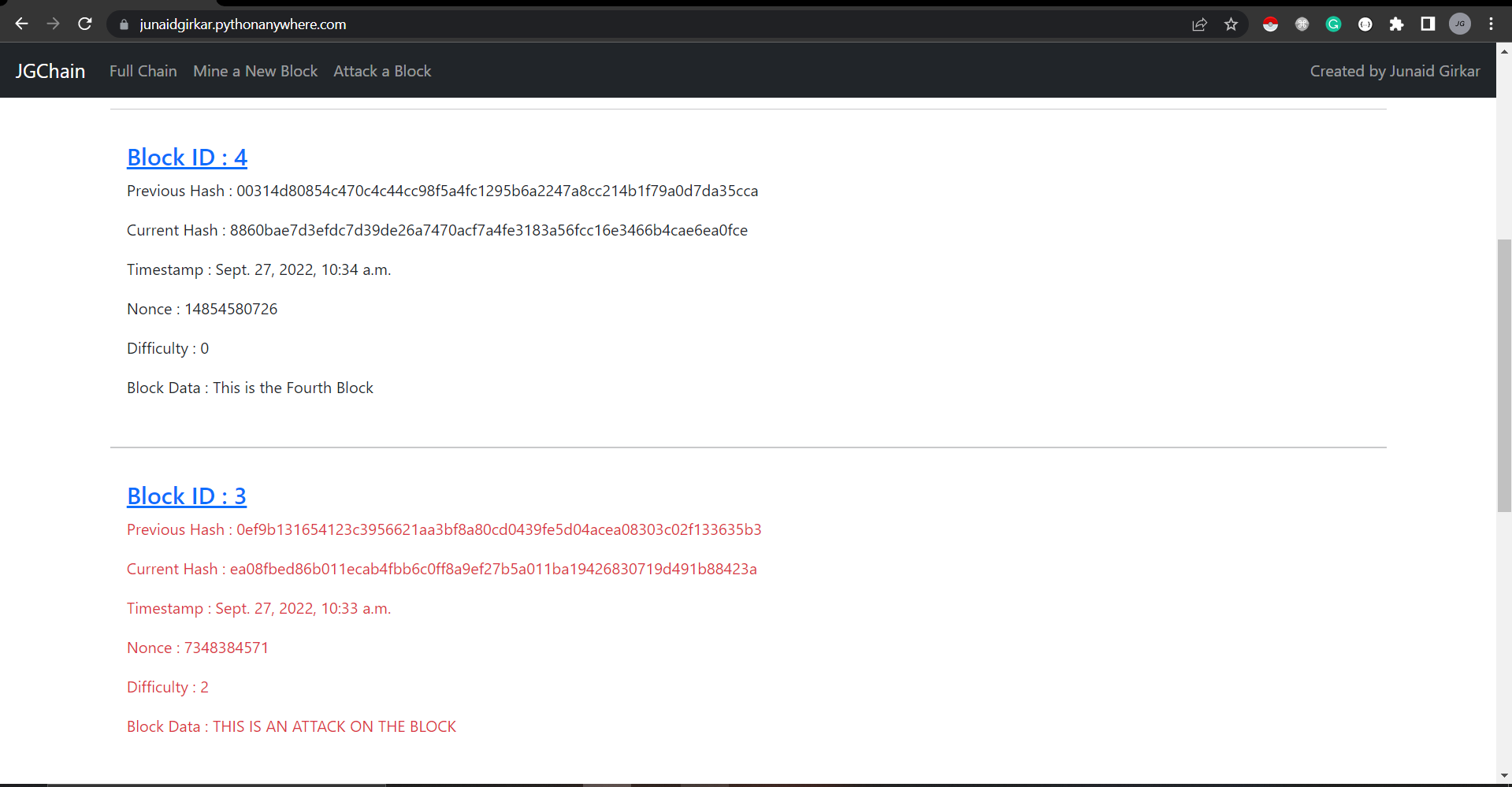
#### Attacking a specific block:



#### After attacking a block:



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



### 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.