01_Blockchain - create a blockchain

using Flask

Code:

```
# module 1- create a Blockchain

# importing the libraries
import datetime
import hashlib
import json
from flask import Flask,jsonify

# Part 1 - Building a Blockchain

class Blockchain:

    def __init__(self):
        # chain in which blocks will be added
        self.chain=[]
        # initialising the first block
        self.create_block(proof=1, previous_hash="0")

    def create_block(self,proof,previous_hash):
        # structure of a block
        block={
                "index":len(self.chain)+1,
                "timestamp":str(datetime.datetime.now()),
                "proof":proof,
                "previous_hash":previous_hash
            }
        # adding the block to the chain
        self.chain.append(block)
        return block

    def get_previous_block(self):
        return self.chain[-1]

    def proof_of_work(self,previous_proof):
        # finding the nonce, through brute force
        new_proof=1
        check_proof=False
        while check_proof is False:
            puzzle=new_proof**2 - previous_proof**2
            hash_operation = hashlib.sha256(str(puzzle).encode()).hexdigest()
            if(hash_operation[:4]=="0000"):
                check_proof=True
            else:
                new_proof+=1
        return new_proof

    def hash(self,block):
        # To get the hash of the block
        encoded_block=json.dumps(block, sort_keys=True).encode()
        return hashlib.sha256(encoded_block).hexdigest()

    def is_chain_valid(self,chain):
        previous_block=chain[0] # Genesis block
        block_index=1
        while block_index<len(chain):
            # Get the current block
            block=chain[block_index]

            # check if the previous hash of the current block doesn't match the hash of the previous block, return False
            if(block["previous_hash"]!=self.hash(previous_block)):
                return False

            # check if the current block's nonce/proof is valid
            proof=block["proof"]
            previous_proof=previous_block["proof"]
            puzzle=proof**2 - previous_proof**2
            hash_operation = hashlib.sha256(str(puzzle).encode()).hexdigest()
            if(hash_operation[:4]!="0000"):
```

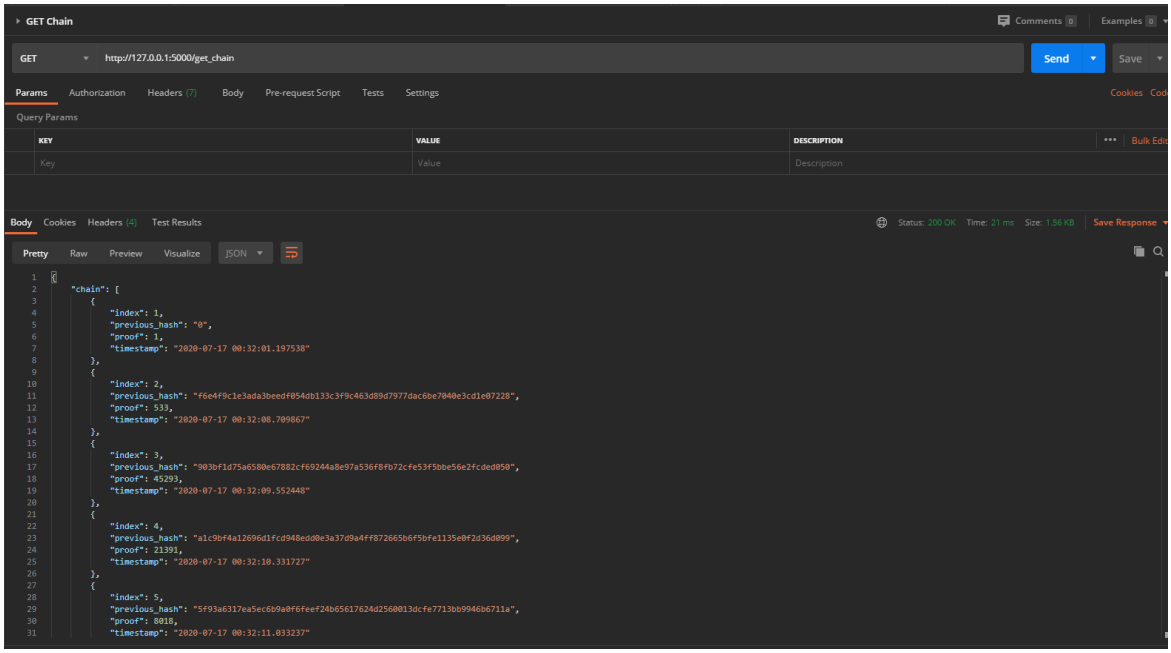
```
                return False

            # make current block as the previous block for the next iteration
            previous_block=block

            # increment the value of block_index
            block_index+=1
        return True

# Part 2 - Mining our blockchain

# Creating a web app
app=Flask(__name__)

# Creating a blockchain
blockchain=Blockchain()

# mining a new block
@app.route("/mine_block",methods=["GET"])
def mine_block():
    # Get the last block as we need its proof/nonce
    previous_block=blockchain.get_previous_block()
    previous_proof=previous_block["proof"];

    # Get the proof/nonce
    proof=blockchain.proof_of_work(previous_proof)

    # Get the previous_hash
    previous_hash=blockchain.hash(previous_block)

    # Create the new block and add it to the chain
    block=blockchain.create_block(proof, previous_hash)

    response={
        **block,
        "message":"Congrats, you just mined a block!",
        }

    return jsonify(response), 200

# Get the full Blockchain
@app.route("/get_chain",methods=["GET"])
def get_chain():
    response={
            "chain": blockchain.chain,
            "length":len(blockchain.chain)
        }
    return jsonify(response), 200

# Get a check result if Blockchain is still valid
@app.route("/is_valid",methods=["GET"])
def is_valid():
    is_valid=blockchain.is_chain_valid(blockchain.chain)
    response={"message":"The Blockchain is NOT valid anymore."}
    if is_valid:
        response={"message":"The Blockchain is valid."}
    return jsonify(response), 200

# Running the app
app.run(host="0.0.0.0",port=5000)
```

Postman collection

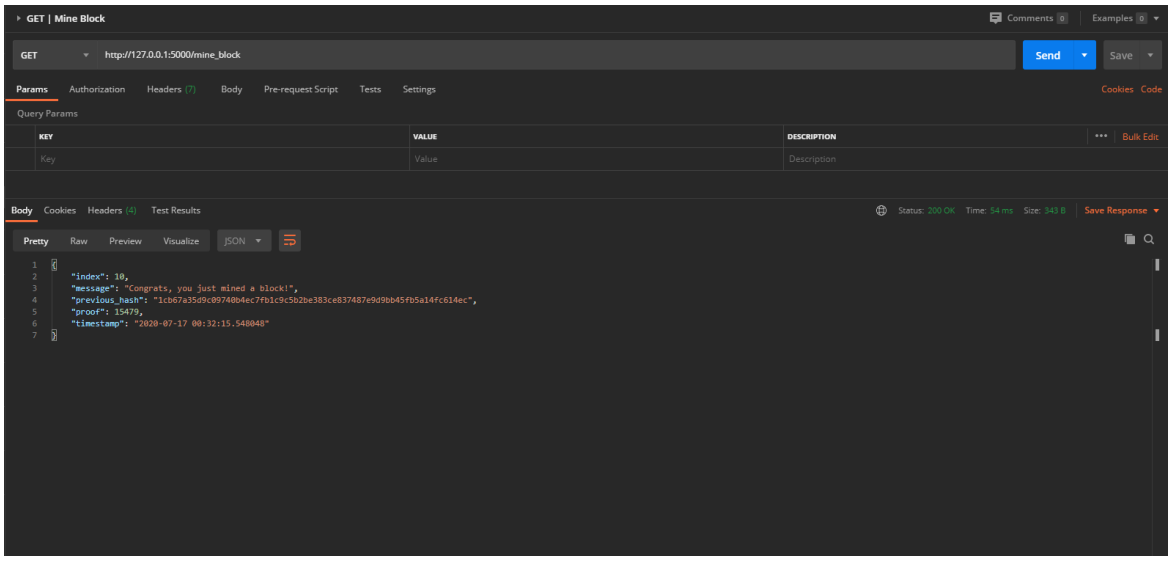
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/403b3f1d-dd28-4edc-93cc-4eb3b13e5c84/Blockchain_A-Z.postman_collection.json

Screenshots:

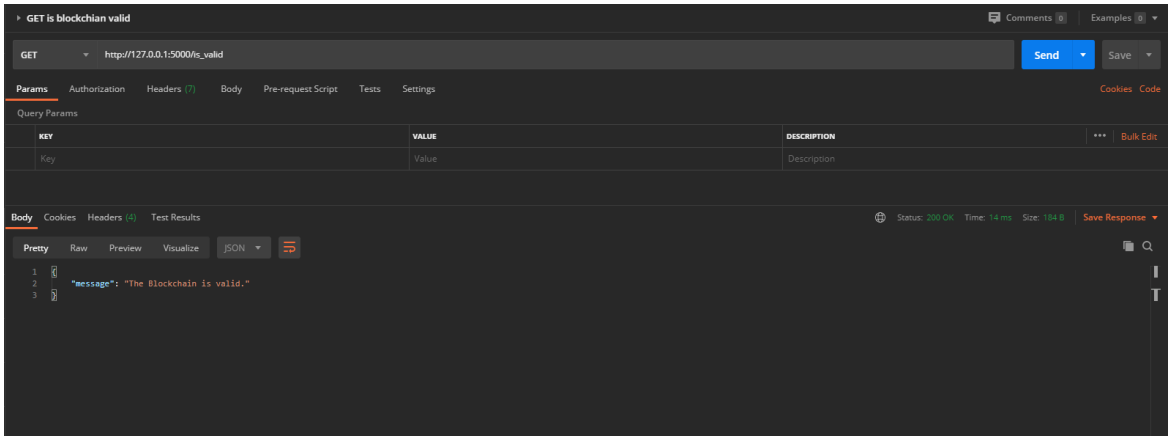
1. GET Chain



2. Mine Block



3. Validate Blockchain


GET is blockchian valid
Comments 0
Examples 0
GET
http://127.0.0.1:5000/is_valid
Send
Save
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 14 ms
Size: 184 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"message": "The Blockchain is valid."
}