Blockchain Journal

Name: MItesh Mahesh Salunkhe

Class: FY MSc Computer Science

Roll no: F003

Sap No: 40772240003

2024

# Python program to create Blockchain

# For timestamp

import datetime

# Calculating the hash

# in order to add digital

# fingerprints to the blocks

import hashlib

# To store data

# in our blockchain

import json

# Flask is for creating the web

# app and jsonify is for

# displaying the blockchain

from flask import Flask, jsonify

class Blockchain:

    # This function is created

    # to create the very first

    # block and set its hash to "0"

    def \_\_init\_\_(self):

        self.chain = []

        self.create\_block(proof=1, previous\_hash='0')

    # This function is created

    # to add further blocks

    # into the chain

    def create\_block(self, proof, previous\_hash):

        block = {'index': len(self.chain) + 1,

                 'timestamp': str(datetime.datetime.now()),

                 'proof': proof,

                 'previous\_hash': previous\_hash}

        self.chain.append(block)

        return block

    # This function is created

    # to display the previous block

    def print\_previous\_block(self):

        return self.chain[-1]

    # This is the function for proof of work

    # and used to successfully mine the block

    def proof\_of\_work(self, previous\_proof):

        new\_proof = 1

        check\_proof = False

        while check\_proof is False:

            hash\_operation = hashlib.sha256(

                str(new\_proof\*\*2 - previous\_proof\*\*2).encode()).hexdigest()

            if hash\_operation[:5] == '00000':

                check\_proof = True

            else:

                new\_proof += 1

        return new\_proof

    def hash(self, block):

        encoded\_block = json.dumps(block, sort\_keys=True).encode()

        return hashlib.sha256(encoded\_block).hexdigest()

    def chain\_valid(self, chain):

        previous\_block = chain[0]

        block\_index = 1

        while block\_index < len(chain):

            block = chain[block\_index]

            if block['previous\_hash'] != self.hash(previous\_block):

                return False

            previous\_proof = previous\_block['proof']

            proof = block['proof']

            hash\_operation = hashlib.sha256(

                str(proof\*\*2 - previous\_proof\*\*2).encode()).hexdigest()

            if hash\_operation[:5] != '00000':

                return False

            previous\_block = block

            block\_index += 1

        return True

# Creating the Web

# App using flask

app = Flask(\_\_name\_\_)

# Create the object

# of the class blockchain

blockchain = Blockchain()

# Mining a new block

@app.route('/mine\_block', methods=['GET'])

def mine\_block():

    previous\_block = blockchain.print\_previous\_block()

    previous\_proof = previous\_block['proof']

    proof = blockchain.proof\_of\_work(previous\_proof)

    previous\_hash = blockchain.hash(previous\_block)

    block = blockchain.create\_block(proof, previous\_hash)

    response = {'message': 'A block is MINED',

                'index': block['index'],

                'timestamp': block['timestamp'],

                'proof': block['proof'],

                'previous\_hash': block['previous\_hash']}

    return jsonify(response), 200

# Display blockchain in json format

@app.route('/get\_chain', methods=['GET'])

def display\_chain():

    response = {'chain': blockchain.chain,

                'length': len(blockchain.chain)}

    return jsonify(response), 200

# Check validity of blockchain

@app.route('/valid', methods=['GET'])

def valid():

    valid = blockchain.chain\_valid(blockchain.chain)

    if valid:

        response = {'message': 'The Blockchain is valid.'}

    else:

        response = {'message': 'The Blockchain is not valid.'}

    return jsonify(response), 200

# Run the flask server locally

app.run(host='127.0.0.1', port=8080)





