EXPERIMENT 2

Name-Kartik Bhat
Class-D20A
Roll No-03

Aim: Create a Blockchain using Python.

Theory:

1)What is Blockchain?
Blockchain is a distributed and decentralized digital ledger used to record transactions across multiple computers in a network. Instead of storing data in a single central database, blockchain stores information in the form of blocks that are linked together in chronological order.
Each block contains transaction data, a cryptographic hash of the previous block, and a timestamp, making the data secure, transparent, and tamper-resistant. Once data is recorded on the blockchain, it cannot be altered without changing all subsequent blocks, which requires network consensus.

2)What is a Block?
A block is the basic unit of a blockchain. It stores a set of verified transactions and links to the previous block using a cryptographic hash.
Blocks are connected sequentially, forming a chain known as the blockchain. The first block in the chain is called the Genesis Block.

3)Components of a Block

Each block in a blockchain typically consists of the following components:

1. **Block Header**
   Contains metadata about the block.It helps in identifying, linking, and validating the block in the blockchain.
   - Previous Block Hash-It stores the hash of the previous block, which links blocks together and ensures that the blockchain cannot be altered.

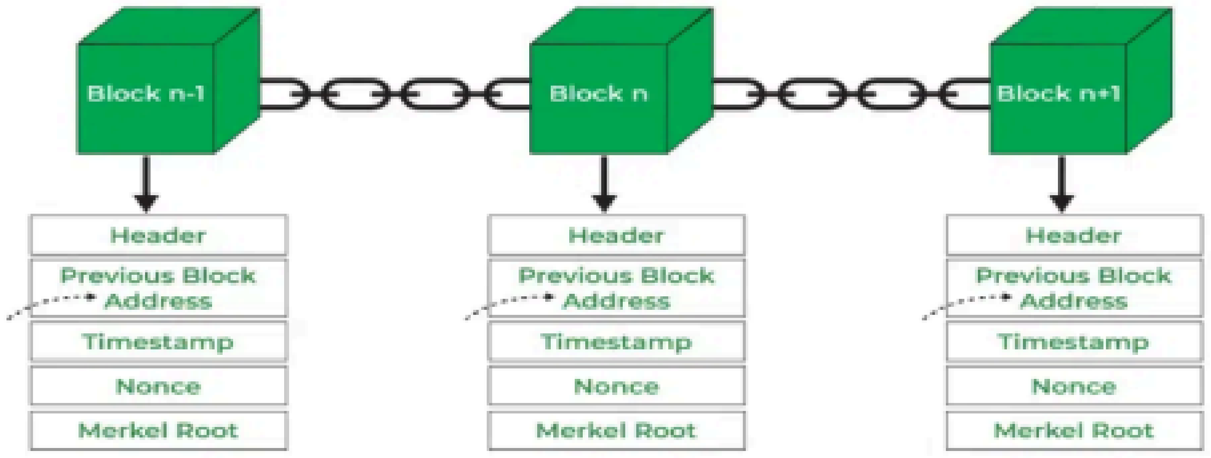
   - Timestamp-The timestamp records the exact date and time when the block was created. It helps maintain the chronological order of blocks in the blockchain.

○ Nonce-A nonce is a random number used during the mining process. Miners change the nonce value to generate a valid hash that meets the network's difficulty level.

○ Merkle Root-The Merkle Root is a single hash value that represents all transactions in the block. It ensures efficient and secure verification of transaction data.

2. **Transaction Data**
Stores all the transactions included in the block.

3. **Hash**
A unique cryptographic value generated using a hashing algorithm (such as SHA-256) that identifies the block and ensures data integrity.



4)Process of Mining

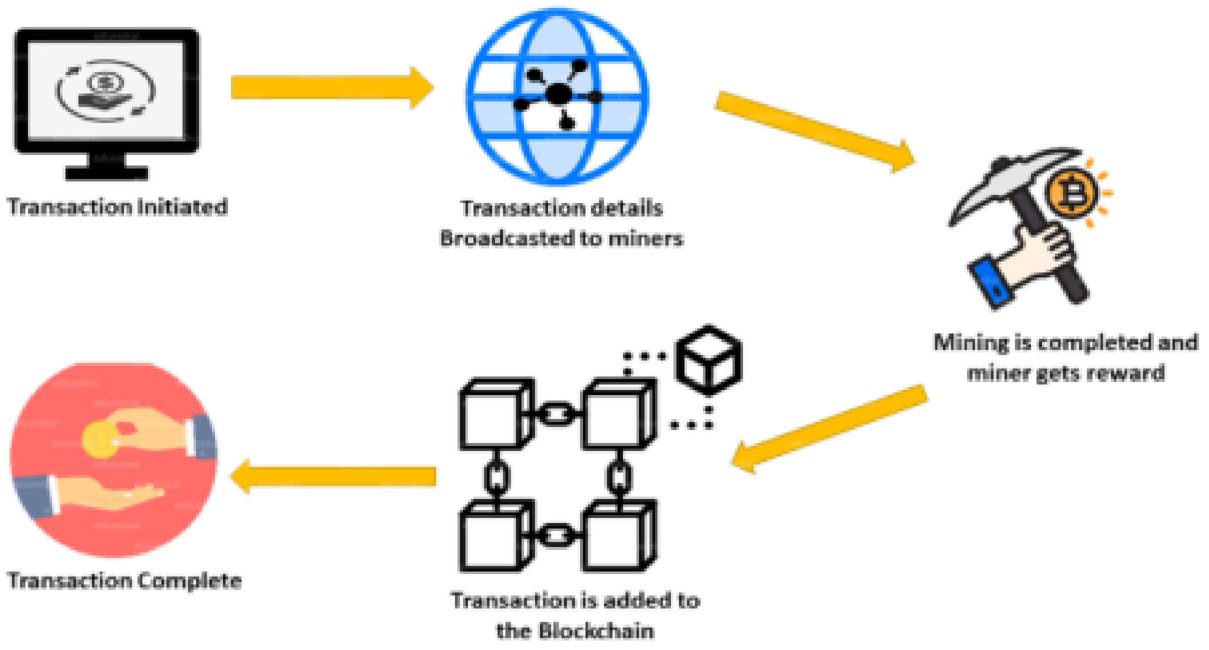
Mining is the process of adding new blocks to the blockchain and validating transactions. It involves solving complex mathematical problems using computational power.

Steps involved in mining:

1. Transactions are collected and verified by miners.

2. Verified transactions are grouped into a block.
3. Miners compete to solve a cryptographic puzzle by finding a valid nonce.
4. The first miner to solve the puzzle broadcasts the block to the network. 5. Other nodes verify the block.
6. Once approved, the block is added to the blockchain, and the miner receives a reward.

Mining ensures security, decentralization, and consensus in the blockchain network.


Transaction Initiated
Transaction details Broadcasted to miners
Mining is completed and miner gets reward
Transaction is added to the Blockchain
Transaction Complete

5)Checking the validity of the blocks in blockchain

The validity of a block is checked using the following methods:

1. Hash Verification
   The block's hash must be correctly calculated and meet the network's difficulty requirements.

2. Previous Hash Matching
   The previous hash stored in the block must match the hash of the preceding block.

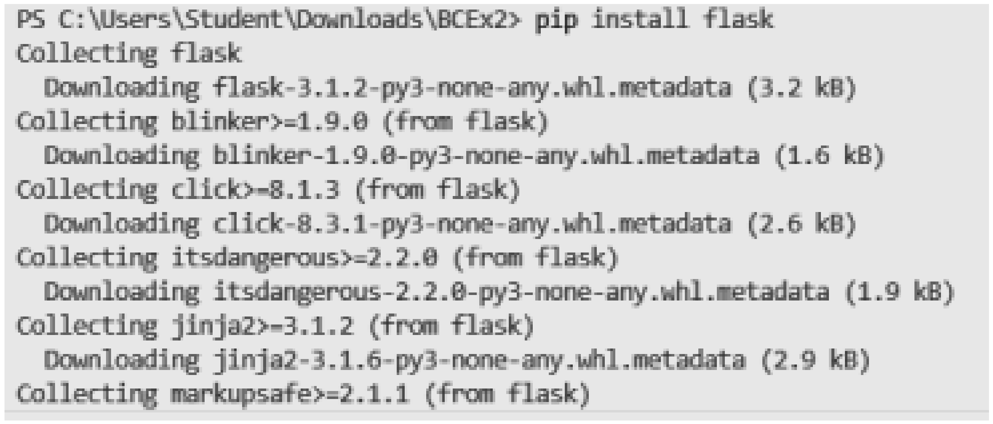
3. Transaction Validation
   All transactions in the block must be valid and digitally signed.

4. Proof of Work Validation
   The nonce must satisfy the cryptographic puzzle defined by the consensus mechanism.

5. Consensus Agreement
   The majority of nodes in the network must agree on the block's validity before it is added to the chain.

Implementation:

Installing Flask



```
PS C:\Users\Student\Downloads\BCEx2> pip install flask
Collecting flask
  Downloading flask-3.1.2-py3-none-any.whl.metadata (3.2 kB)
Collecting blinker>=1.9.0 (from flask)
  Downloading blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
Collecting click>=8.1.3 (from flask)
  Downloading click-8.3.1-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.2.0 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting jinja2>=3.1.2 (from flask)
  Downloading jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Collecting markupsafe>=2.1.1 (from flask)
```

blockchain.py

```
from flask import Flask, jsonify

import datetime

import hashlib

import json



# ---------------------------------------------

# Blockchain Class
# ----------------------------------------------
class Blockchain:

    def __init__(self):
        self.chain = []
        self.transactions = []
        # Create Genesis Block
        self.create_block(proof=1, previous_hash='0')

    # Create a new block
    def create_block(self, proof, previous_hash):
        block = {
            'index': len(self.chain) + 1,
            'timestamp': str(datetime.datetime.now()),
            'transactions': self.transactions,
            'proof': proof,
            'previous_hash': previous_hash
        }

        # Remove transactions after mining
    self.transactions = []

    self.chain.append(block)

    return block



# Get previous block

def get_previous_block(self):

    return self.chain[-1]



# Create a new transaction

def create_transaction(self, sender, receiver, amount):

    self.transactions.append({

        'sender': sender,

        'receiver': receiver,

        'amount': amount

    })

    return self.get_previous_block()['index'] + 1



# Proof of Work (Golden Nonce with "000")

def proof_of_work(self, previous_proof):

    new_proof = 1
    check_proof = False

    while not check_proof:
        hash_operation = hashlib.sha256(
            str(new_proof**2 - previous_proof**2).encode()
        ).hexdigest()

        # Cryptographic puzzle: leading "000"
        if hash_operation[:3] == "000":
            check_proof = True
        else:
            new_proof += 1

    return new_proof

# Hash a block
def hash(self, block):
    encoded_block = json.dumps(block, sort_keys=True).encode()
    return hashlib.sha256(encoded_block).hexdigest()
```

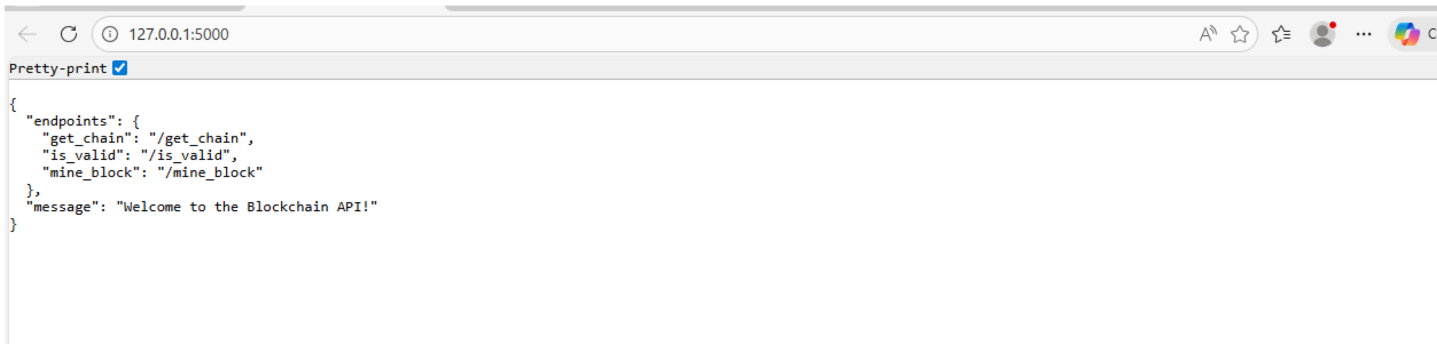
```
# Validate the blockchain

def is_chain_valid(self, chain):

    previous_block = chain[0]

    block_index = 1



    while block_index < len(chain):

        block = chain[block_index]



        # Check previous hash

        if block['previous_hash'] != self.hash(previous_block):

            return False



        # Check Proof of Work

        previous_proof = previous_block['proof']

        proof = block['proof']

        hash_operation = hashlib.sha256(

            str(proof**2 - previous_proof**2).encode()

        ).hexdigest()



        if hash_operation[:3] != "000":
                return False

            previous_block = block
            block_index += 1

        return True


# ------------------------------------------------
# Flask Web App
# ------------------------------------------------
app = Flask(__name__)
blockchain = Blockchain()

# Mine a new block (only if transactions exist)
@app.route('/mine_block', methods=['GET'])
def mine_block():
    if len(blockchain.transactions) == 0:
        return jsonify({'message': 'No transactions to mine'}), 400
    previous_block = blockchain.get_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': 'Block mined successfully!',
        'block': block
    }
    return jsonify(response), 200

# Add a transaction
@app.route('/add_transaction', methods=['GET'])
def add_transaction():
    blockchain.create_transaction(
        sender="Alice",
        receiver="Bob",
        amount=50
    )
    return jsonify({'message': 'Transaction added successfully'}), 201


# Get full blockchain

@app.route('/get_chain', methods=['GET'])

def get_chain():

    response = {

        'chain': blockchain.chain,

        'length': len(blockchain.chain)

    }

    return jsonify(response), 200


# Validate blockchain

@app.route('/is_valid', methods=['GET'])

def is_valid():

    is_valid = blockchain.is_chain_valid(blockchain.chain)

    if is_valid:

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

    else:

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

    return jsonify(response), 200
```

```
# Run the app

if __name__ == '__main__':

    app.run(debug=True)
```

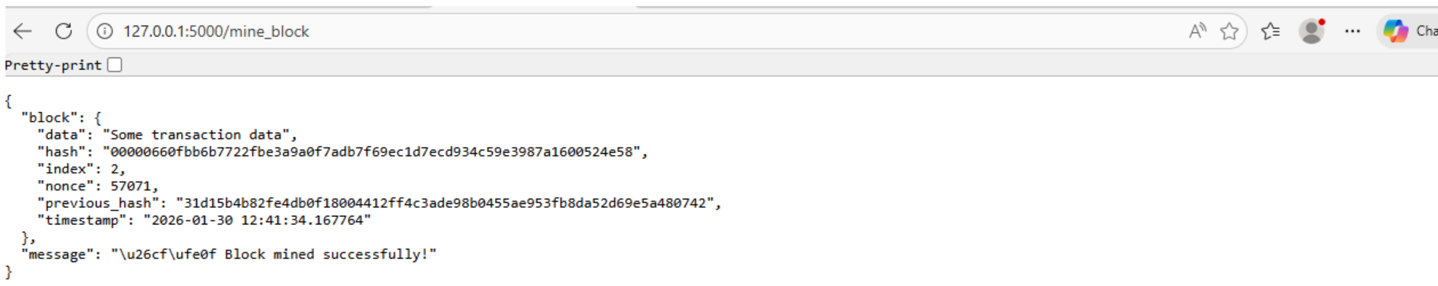
app.py

```
from flask import Flask, jsonify
from blockchain import Blockchain

app = Flask(__name__)
blockchain = Blockchain()

@app.route('/')
def index():
    return jsonify({
        'message': 'Welcome to the Blockchain API!',
        'endpoints': {
            'get_chain': '/get_chain',
            'mine_block': '/mine_block',
            'is_valid': '/is_valid'
        }
    })

@app.route('/mine_block', methods=['GET'])
def mine_block():
    block = blockchain.mine_block("Some
transaction data")
```

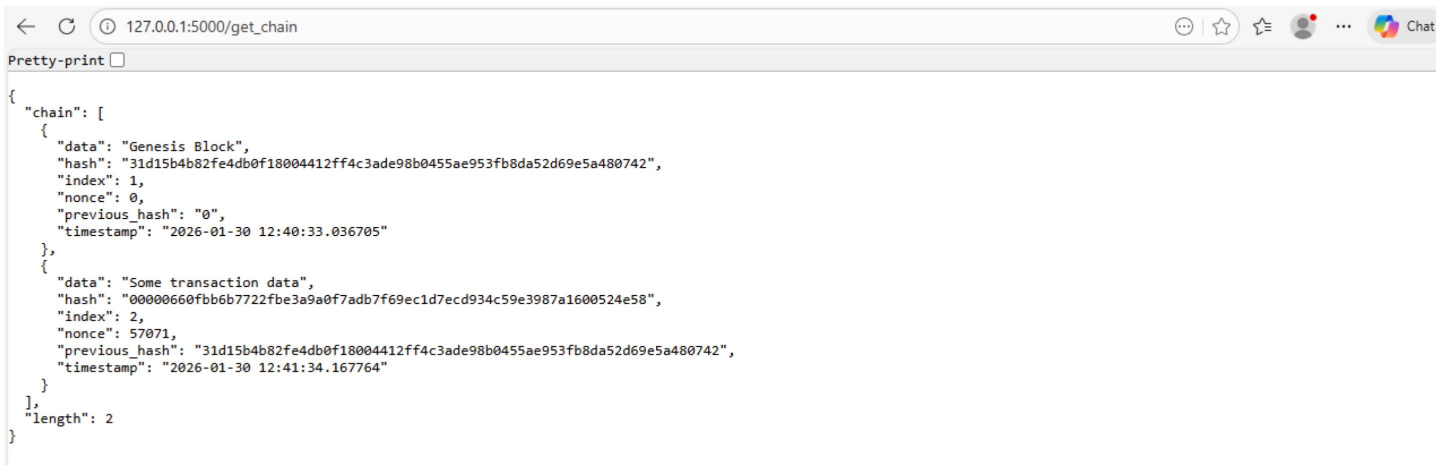
```
    return jsonify({
        'message': '⛏️ Block mined successfully!',
        'block': block
    }), 200

@app.route('/get_chain', methods=['GET'])
def get_chain():
    return jsonify({
        'chain': blockchain.chain,
        'length': len(blockchain.chain)
    }), 200

@app.route('/is_valid', methods=['GET'])
def is_valid():
    return jsonify({
        'is_valid': blockchain.is_chain_valid()
    }), 200

if __name__ == '__main__':
    print(" Mining blockchain with 4 leading
zeroes...")
    app.run(debug=True, use_reloader=False)
```



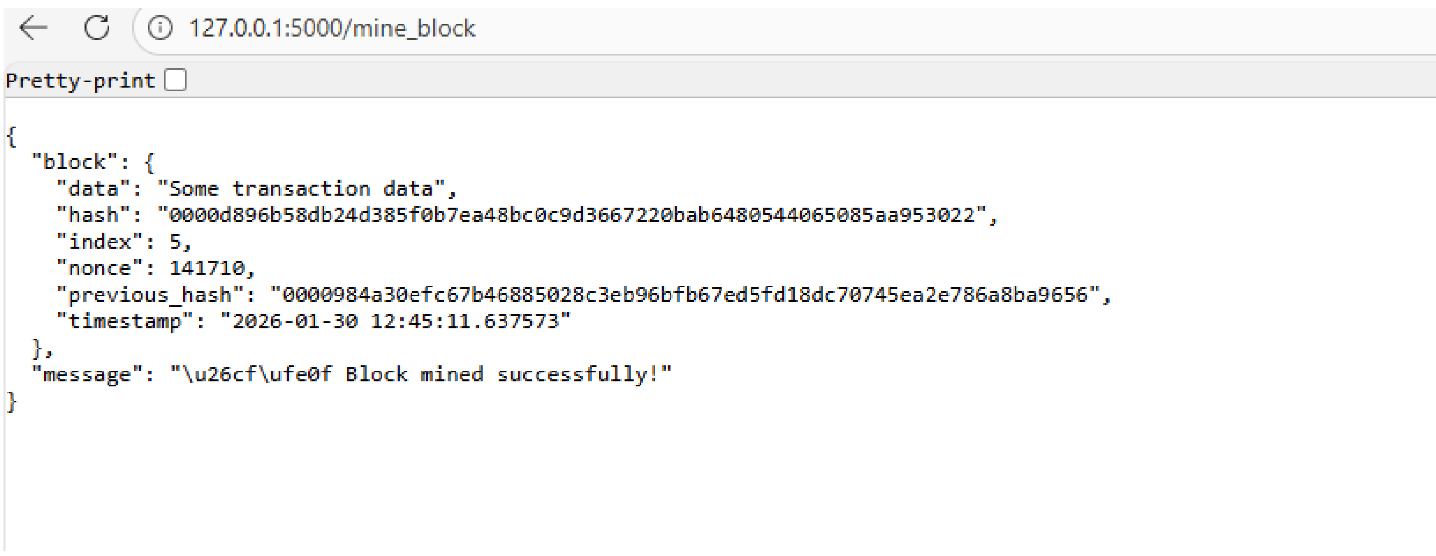
```
{
  "endpoints": {
    "get_chain": "/get_chain",
    "is_valid": "/is_valid",
    "mine_block": "/mine_block"
  },
  "message": "Welcome to the Blockchain API!"
}
```



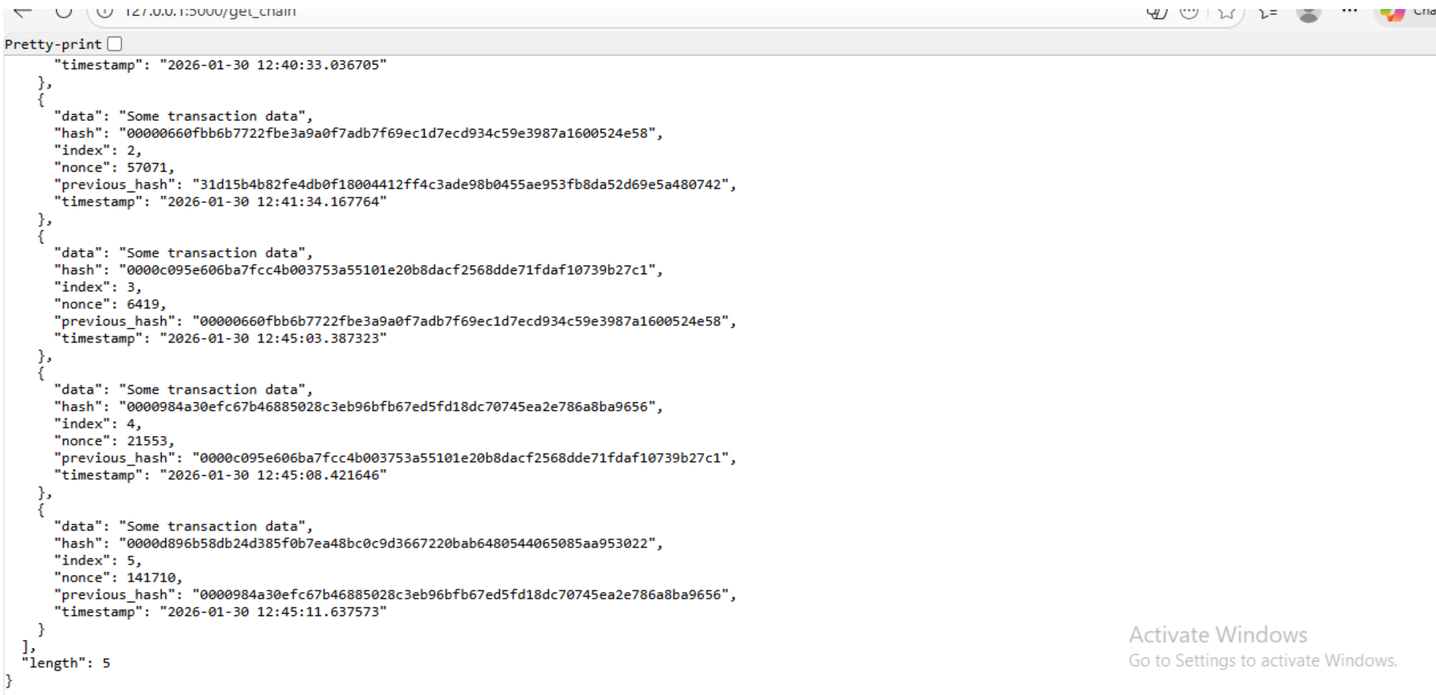
```
{
  "block": {
    "data": "Some transaction data",
    "hash": "00000660fbb6b7722fbe3a9a0f7adb7f69ec1d7ecd934c59e3987a1600524e58",
    "index": 2,
    "nonce": 57071,
    "previous_hash": "31d15b4b82fe4db0f18004412ff4c3ade98b0455ae953fb8da52d69e5a480742",
    "timestamp": "2026-01-30 12:41:34.167764"
  },
  "message": "\u26cf\ufe0f Block mined successfully!"
}
```



```
{
  "chain": [
    {
      "data": "Genesis Block",
      "hash": "31d15b4b82fe4db0f18004412ff4c3ade98b0455ae953fb8da52d69e5a480742",
      "index": 1,
      "nonce": 0,
      "previous_hash": "0",
      "timestamp": "2026-01-30 12:40:33.036705"
    },
    {
      "data": "Some transaction data",
      "hash": "00000660fbb6b7722fbe3a9a0f7adb7f69ec1d7ecd934c59e3987a1600524e58",
      "index": 2,
      "nonce": 57071,
      "previous_hash": "31d15b4b82fe4db0f18004412ff4c3ade98b0455ae953fb8da52d69e5a480742",
      "timestamp": "2026-01-30 12:41:34.167764"
    }
  ],
  "length": 2
}
```



```
{
  "block": {
    "data": "Some transaction data",
    "hash": "0000d896b58db24d385f0b7ea48bc0c9d3667220bab6480544065085aa953022",
    "index": 5,
    "nonce": 141710,
    "previous_hash": "0000984a30efc67b46885028c3eb96bfb67ed5fd18dc70745ea2e786a8ba9656",
    "timestamp": "2026-01-30 12:45:11.637573"
  },
  "message": "\u26cf\ufe0f Block mined successfully!"
}
```

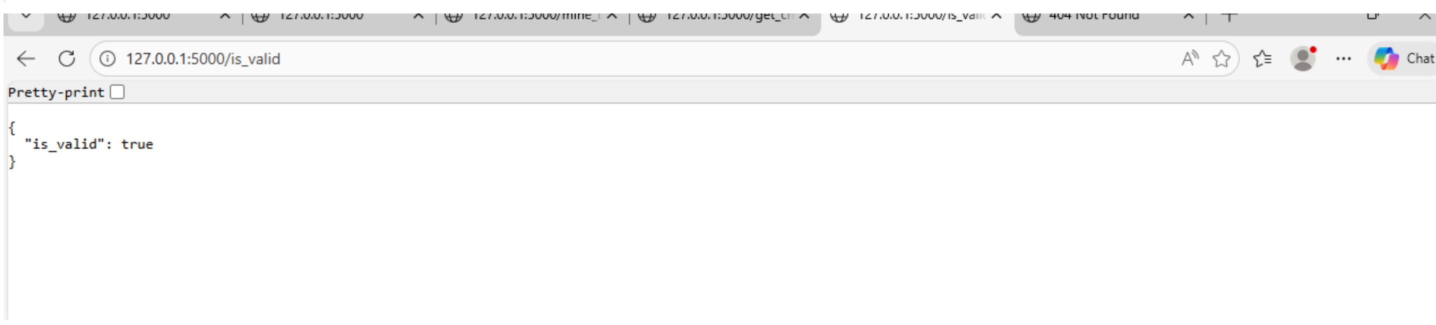


127.0.0.1:5000/get_chain

Pretty-print ☐

```
      "timestamp": "2026-01-30 12:40:33.036705"
    },
    {
      "data": "Some transaction data",
      "hash": "00000660fbb6b7722fbe3a9a0f7adb7f69ec1d7ecd934c59e3987a1600524e58",
      "index": 2,
      "nonce": 57071,
      "previous_hash": "31d15b4b82fe4db0f18004412ff4c3ade98b0455ae953fb8da52d69e5a480742",
      "timestamp": "2026-01-30 12:41:34.167764"
    },
    {
      "data": "Some transaction data",
      "hash": "0000c095e606ba7fcc4b003753a55101e20b8dacf2568dde71fdaf10739b27c1",
      "index": 3,
      "nonce": 6419,
      "previous_hash": "00000660fbb6b7722fbe3a9a0f7adb7f69ec1d7ecd934c59e3987a1600524e58",
      "timestamp": "2026-01-30 12:45:03.387323"
    },
    {
      "data": "Some transaction data",
      "hash": "0000984a30efc67b46885028c3eb96bfb67ed5fd18dc70745ea2e786a8ba9656",
      "index": 4,
      "nonce": 21553,
      "previous_hash": "0000c095e606ba7fcc4b003753a55101e20b8dacf2568dde71fdaf10739b27c1",
      "timestamp": "2026-01-30 12:45:08.421646"
    },
    {
      "data": "Some transaction data",
      "hash": "0000d896b58db24d385f0b7ea48bc0c9d3667220bab6480544065085aa953022",
      "index": 5,
      "nonce": 141710,
      "previous_hash": "0000984a30efc67b46885028c3eb96bfb67ed5fd18dc70745ea2e786a8ba9656",
      "timestamp": "2026-01-30 12:45:11.637573"
    }
  ],
  "length": 5
}
```

Activate Windows
Go to Settings to activate Windows.



127.0.0.1:5000/is_valid

Pretty-print ☐

```
{
  "is_valid": true
}
```