Ria Chaudhari D20A14

BLOCKCHAIN LAB EXP 2

AIM: Create a Blockchain using Python

THEORY:

1.What is Blockchain?

Blockchain is a decentralized and distributed digital ledger used to record transactions across many computers in a secure and tamper-proof manner. The data is stored in units called blocks, and each block is connected to the previous one using cryptographic hashes, forming a chain. Every block contains transaction data, its own hash, and the hash of the previous block, which ensures integrity and transparency. Once information is added to the blockchain, it becomes extremely difficult to modify, making the system reliable and trustworthy. Blockchain technology is widely used in cryptocurrencies, supply chain management, healthcare, and many other applications where secure record keeping is required.

2.Process of Mining

Mining is the process of adding a new block to the blockchain.Miners compete to solve a mathematical puzzle (Proof of Work).

Steps:

1. Transactions are broadcast to the network.
2. Miners collect them into a block.
3. They try to find a **nonce** value.
4. When nonce is combined with block data → it must produce a hash that satisfies a rule (example: starts with `0000`).
5. First miner who finds it → broadcasts the solution.
6. Other nodes verify it.
7. If valid → block is added → miner gets reward.

Mining provides: Security, Consensus,New coin generation

3.How to Check the Validity of Blocks in a Blockchain

To verify blockchain integrity, nodes check mainly three things:

1. Hash correctness

Recalculate the block hash and compare it with the stored hash.

If different → block is invalid.

2. Previous hash link

The block's `previous_hash` must match the hash of the previous block.

If not → chain is broken.

3. Proof of Work validity

Check whether the hash satisfies the difficulty rule
(example: starts with required number of zeros).

CODE:

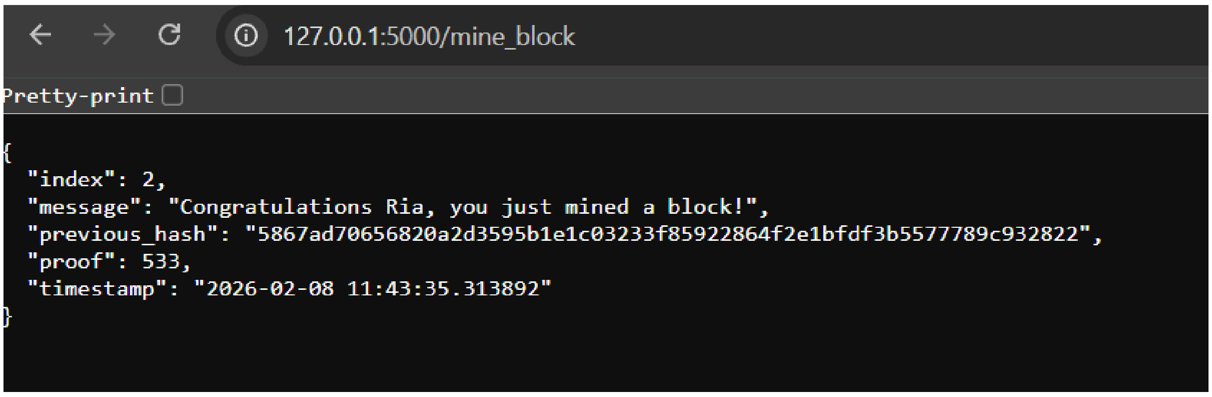
```
# Importing libraries

import datetime # To generate timestamps for blocks

import hashlib # To generate SHA-256 hashes

import json # To convert block data into JSON format

from flask import Flask, jsonify # To create REST API endpoints

# Part 1 - Building the Blockchain

class Blockchain:

def __init__(self):

# List to store the blockchain

self.chain = []

# Create the genesis block

self.create_block(proof=1, previous_hash='0')

def create_block(self, proof, previous_hash):

"""

Create a new block and add it to the blockchain

"""

block = {

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

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

'proof': proof,

'previous_hash': previous_hash

}

self.chain.append(block)
```

```
return block
def get_previous_block(self):
"""
Return the last block in the chain
"""
return self.chain[-1]
def proof_of_work(self, previous_proof):
"""
Proof of Work Algorithm:
Find a number such that the hash of
(new_proof^2 - previous_proof^2)
starts with '0000'
"""
new_proof = 1
check_proof = False
while not check_proof:
hash_operation = hashlib.sha256(
str(new_proof**2 - previous_proof**2).encode()
).hexdigest()
if hash_operation[:4] == '0000':
check_proof = True
else:
new_proof += 1
return new_proof
def hash(self, block):
"""
```

```
Create a SHA-256 hash of a block
"""
encoded_block = json.dumps(block, sort_keys=True).encode()
return hashlib.sha256(encoded_block).hexdigest()
def is_chain_valid(self, chain):
"""
Check whether the blockchain is valid
"""
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[:4] != '0000':
return False
previous_block = block
block_index += 1
return True
```

```
# Part 2 - Mining the Blockchain (Flask API)

# Create Flask app

app = Flask(__name__)

# Create Blockchain object

blockchain = Blockchain()

# API Endpoint - Mine a new block

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

def mine_block():

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': 'Congratulations Ria, you just mined a block!',

'index': block['index'],

'timestamp': block['timestamp'],

'proof': block['proof'],

'previous_hash': block['previous_hash']

}

return jsonify(response), 200

# API Endpoint - Get full blockchain

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

def get_chain():

response = {

'chain': blockchain.chain,
'length': len(blockchain.chain)

}

return jsonify(response), 200

# API Endpoint - Check blockchain validity

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

def is_valid():

valid = blockchain.is_chain_valid(blockchain.chain)

if valid:

response = {'message': 'Blockchain is valid, Ria.'}

else:

response = {'message': 'Blockchain is NOT valid, Ria.'}

return jsonify(response), 200

# Run the Flask Application

if __name__ == '__main__':

app.run(host='0.0.0.0', port=5000)
```

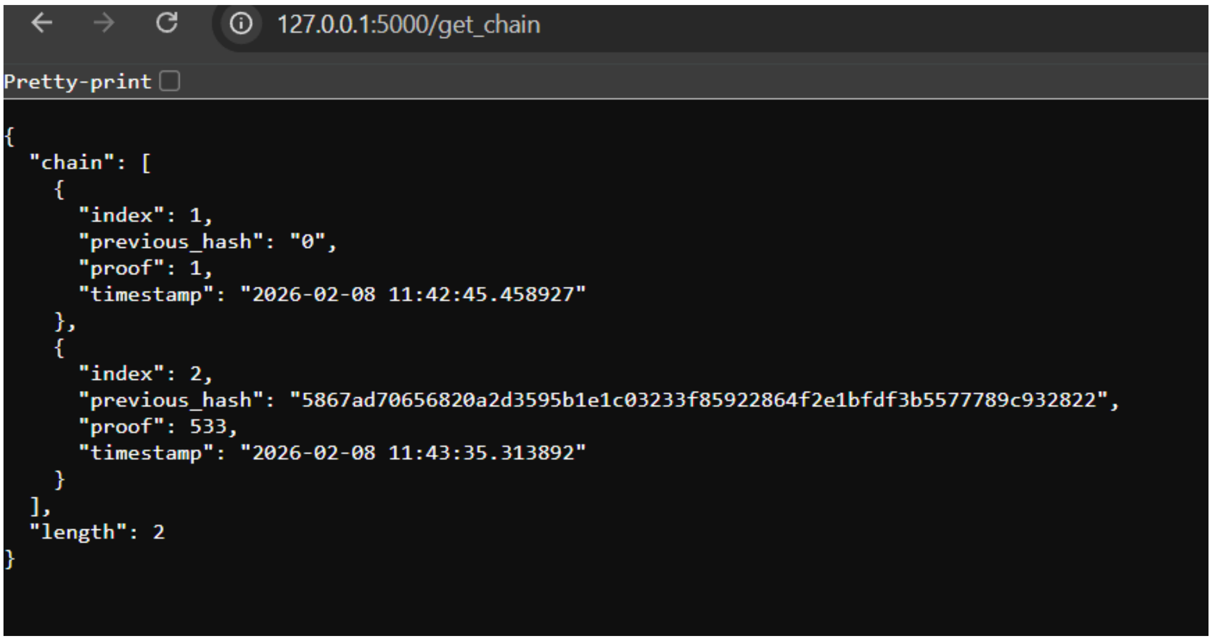
OUTPUT:



127.0.0.1:5000/mine_block

Pretty-print

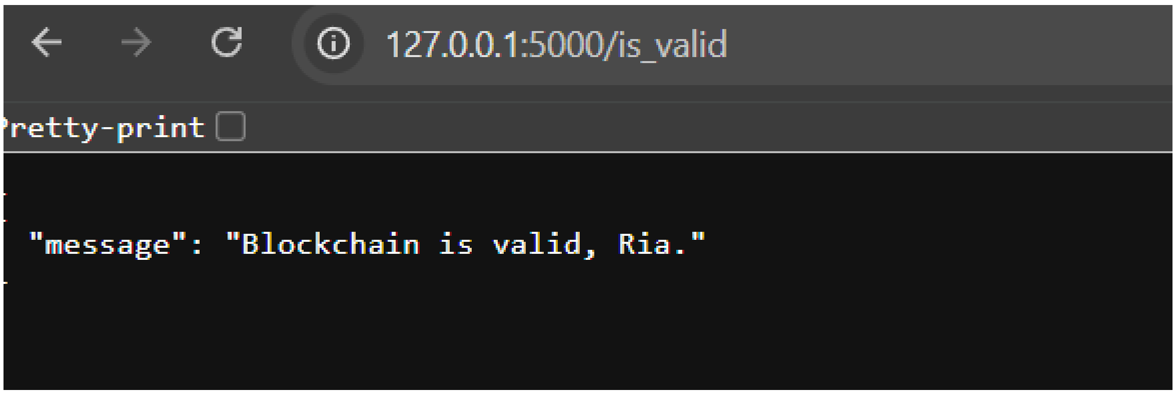
```
{
  "index": 2,
  "message": "Congratulations Ria, you just mined a block!",
  "previous_hash": "5867ad70656820a2d3595b1e1c03233f85922864f2e1bfdf3b5577789c932822",
  "proof": 533,
  "timestamp": "2026-02-08 11:43:35.313892"
}
```



127.0.0.1:5000/get_chain

Pretty-print

```
{
  "chain": [
    {
      "index": 1,
      "previous_hash": "0",
      "proof": 1,
      "timestamp": "2026-02-08 11:42:45.458927"
    },
    {
      "index": 2,
      "previous_hash": "5867ad70656820a2d3595b1e1c03233f85922864f2e1bfdf3b5577789c932822",
      "proof": 533,
      "timestamp": "2026-02-08 11:43:35.313892"
    }
  ],
  "length": 2
}
```



127.0.0.1:5000/is_valid

retty-print

```
  "message": "Blockchain is valid, Ria."
```

CONCLUSION:
In this experiment, a simple blockchain was successfully implemented using Python and Flask. The system allows block mining, viewing the complete blockchain, and checking its validity. Proof-of-Work was used to make block creation secure, while cryptographic hashing ensured that the data stored in blocks remains unchanged. This implementation helps in understanding the basic concepts of blockchain such as immutability, consensus, and decentralization, and provides a good foundation for learning advanced blockchain applications.