Name- Diksha Utekar D20A /06

Experiment No: 2

Aim: Create a Blockchain using Python

Theory:

Blockchain is a distributed and immutable ledger that records data in a sequence of linked blocks. Each block contains a timestamp, proof (from Proof-of-Work), and the hash of the previous block, ensuring integrity and security.

In this program:

1. **Block Creation** – A block is generated using `create_block()` with a proof and previous hash.

2. **Proof-of-Work (PoW)** – Implemented to mine a block by solving a cryptographic puzzle (finding a hash with leading zeros).

3. **Hashing** – SHA-256 algorithm ensures data immutability.

4. **Validation** – `is_chain_valid()` checks that every block correctly references the previous hash and satisfies PoW conditions.

5. **Flask Web API** – Routes `/mine_block`, `/get_chain`, and `/is_valid` allow mining, fetching the chain, and verifying blockchain integrity through a browser or API client.

This program runs on a local server and simulates mining a blockchain without a network of nodes, making it an ideal introductory model to understand blockchain basics.

Code Description:

1. Importing Required Libraries

- **datetime** → to timestamp each block

- **hashlib** → to generate SHA-256 hashes

- **json** → to encode blocks as JSON for hashing

- **Flask** → to run the blockchain API serve

2. Blockchain Class – Core Logic

This contains everything needed to:

- create blocks
- store the chain
- implement Proof of Work
- validate chain integrity

3. Flask Web API

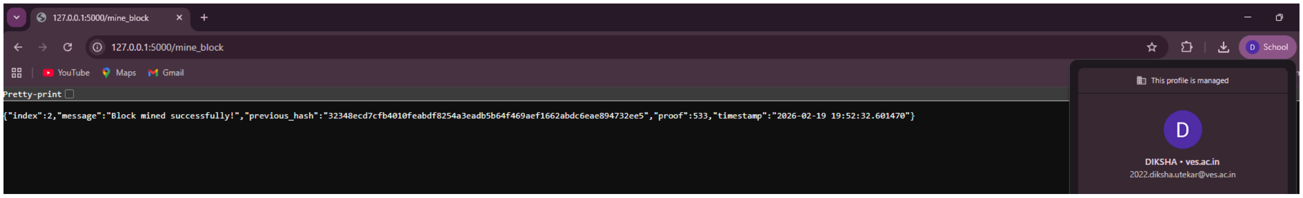
Provides endpoints:

- `/mine_block` → mine a new block
- `/get_chain` → fetch full blockchain
- `/is_valid` → verify blockchain

CODE:

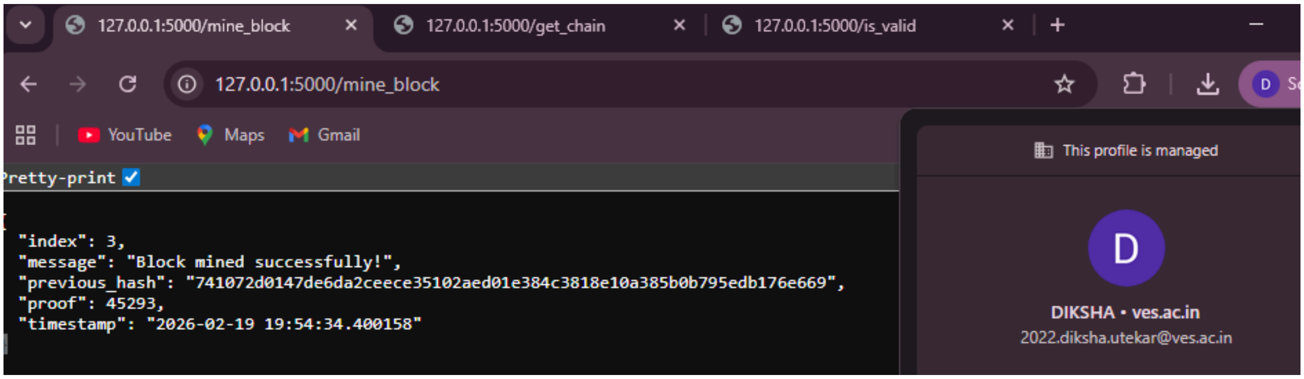
```
# Required installation:
pip install flask==2.2.5

import datetime
import hashlib
import json
from flask import Flask, jsonify


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

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

    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

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

    def proof_of_work(self, previous_proof):
        new_proof = 1
        while True:
            hash_operation = hashlib.sha256(
               str(new_proof**2 - previous_proof**2).encode()
            ).hexdigest()
            if hash_operation[:4] == '0000':
                return new_proof
            new_proof += 1

    def hash(self, 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]
        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[:4] != '0000':
                return False

            previous_block = block
            block_index += 1

        return True


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


@app.route('/mine_block',
methods=['GET']) def mine_block():
    previous_block = blockchain.get_previous_block()
    proof = blockchain.proof_of_work(previous_block['proof'])
    previous_hash = blockchain.hash(previous_block)
    block = blockchain.create_block(proof, previous_hash)

    response = {
        'message': 'Block mined successfully!',
        'index': block['index'],
        'timestamp': block['timestamp'],
        'proof': block['proof'],
        'previous_hash':
        block['previous_hash']
    }
    return jsonify(response), 200


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


@app.route('/is_valid', methods=['GET'])
def is_valid():
    if blockchain.is_chain_valid(blockchain.chain):
        response = {'message': 'Blockchain is valid.'}
    else:
        response = {'message': 'Blockchain is not valid.'}
    return jsonify(response), 200


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

Output:


127.0.0.1:5000/mine_block
127.0.0.1:5000/mine_block
YouTube
Maps
Gmail
School
This profile is managed
D
DIKSHA • ves.ac.in
2022.diksha.utekar@ves.ac.in

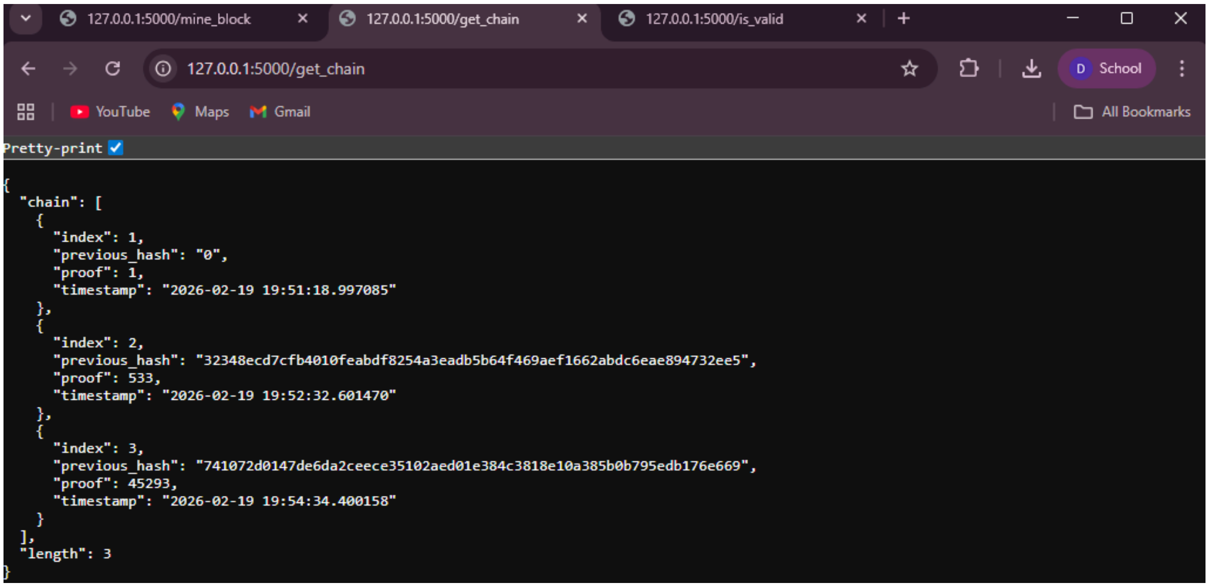
Pretty-print □

```
{"index":2,"message":"Block mined successfully!","previous_hash":"32348ecd7cfb4010feabdf8254a3eadb5b64f469aef1662abdc6eae894732ee5","proof":533,"timestamp":"2026-02-19 19:52:32.601470"}
```



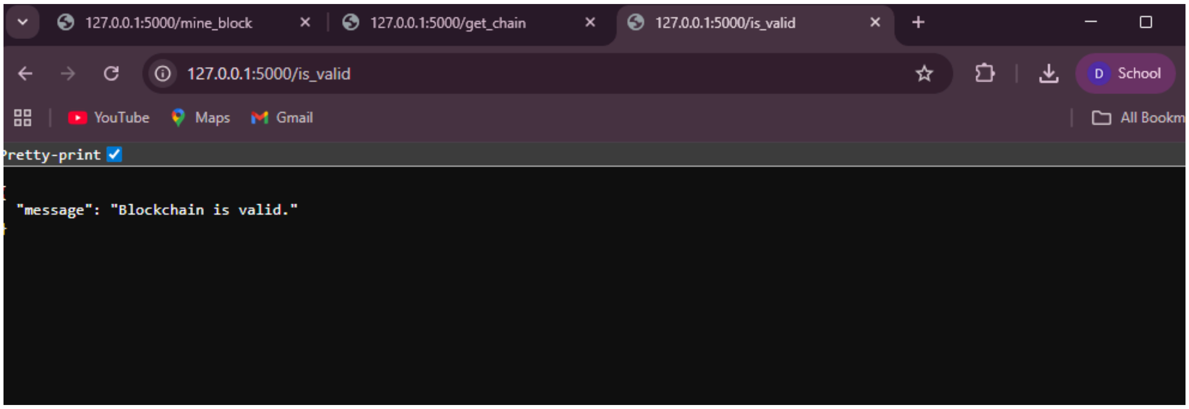
Pretty-print ☑

```
{
  "index": 3,
  "message": "Block mined successfully!",
  "previous_hash": "741072d0147de6da2ceece35102aed01e384c3818e10a385b0b795edb176e669",
  "proof": 45293,
  "timestamp": "2026-02-19 19:54:34.400158"
}
```



Pretty-print ☑

```
{
  "chain": [
    {
      "index": 1,
      "previous_hash": "0",
      "proof": 1,
      "timestamp": "2026-02-19 19:51:18.997085"
    },
    {
      "index": 2,
      "previous_hash": "32348ecd7cfb4010feabdf8254a3eadb5b64f469aef1662abdc6eae894732ee5",
      "proof": 533,
      "timestamp": "2026-02-19 19:52:32.601470"
    },
    {
      "index": 3,
      "previous_hash": "741072d0147de6da2ceece35102aed01e384c3818e10a385b0b795edb176e669",
      "proof": 45293,
      "timestamp": "2026-02-19 19:54:34.400158"
    }
  ],
  "length": 3
}
```



Pretty-print ☑

```
{
  "message": "Blockchain is valid."
}
```

Conclusion:

We successfully created a basic blockchain using Python and Flask that supports block mining, chain retrieval, and validity checks. Proof-of-Work ensures security by making block creation computationally intensive, while cryptographic hashing guarantees immutability. The implementation demonstrates core blockchain principles—decentralization, immutability, and consensus—in a simplified environment, providing a strong foundation for building more advanced blockchain systems.