Blockchain Lab Exp.2

Name: Ronit Santwani **Class:** D20A
Roll No: 63

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.

1.What is a Blockchain?

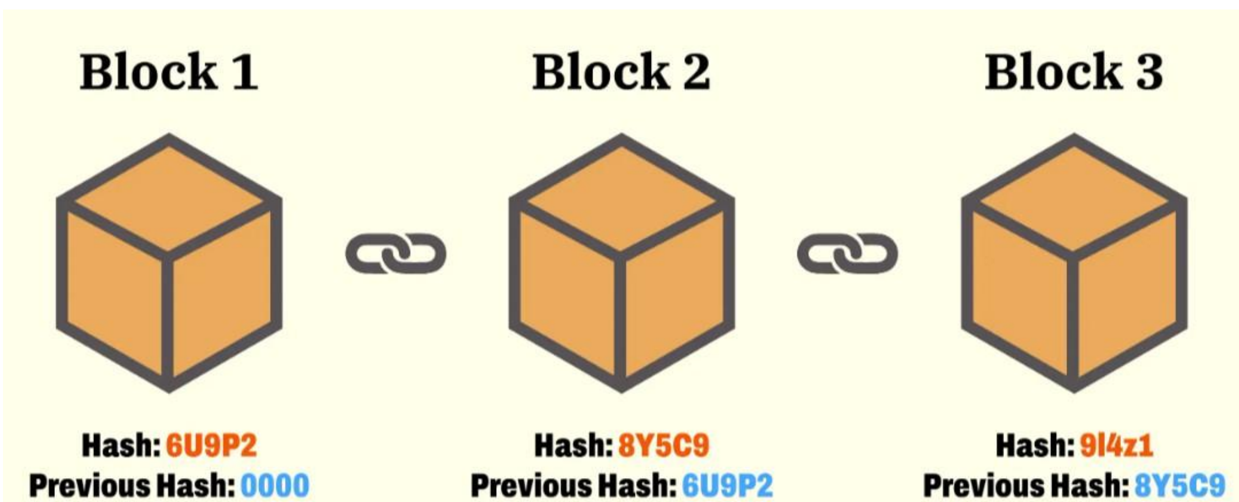
A **blockchain** is a **distributed, decentralized, and immutable digital ledger** used to store data securely across multiple computers (nodes). Instead of relying on a central authority, blockchain technology allows participants in a network to **collectively maintain and verify records**.

The data in a blockchain is stored in units called **blocks**. Each block contains:

- Data or transactions
- Timestamp
- A cryptographic hash of the previous block
- A proof value generated through a consensus mechanism

Blocks are linked together using **cryptographic hash functions**, forming a continuous chain. Once a block is added, it **cannot be modified or deleted** without altering all subsequent blocks, which makes the blockchain highly secure. This property is known as **immutability**.

Blockchain technology provides **transparency**, **security**, and **trust**, and is widely used in applications such as cryptocurrencies (Bitcoin), supply chain management, healthcare records, and digital identity systems.



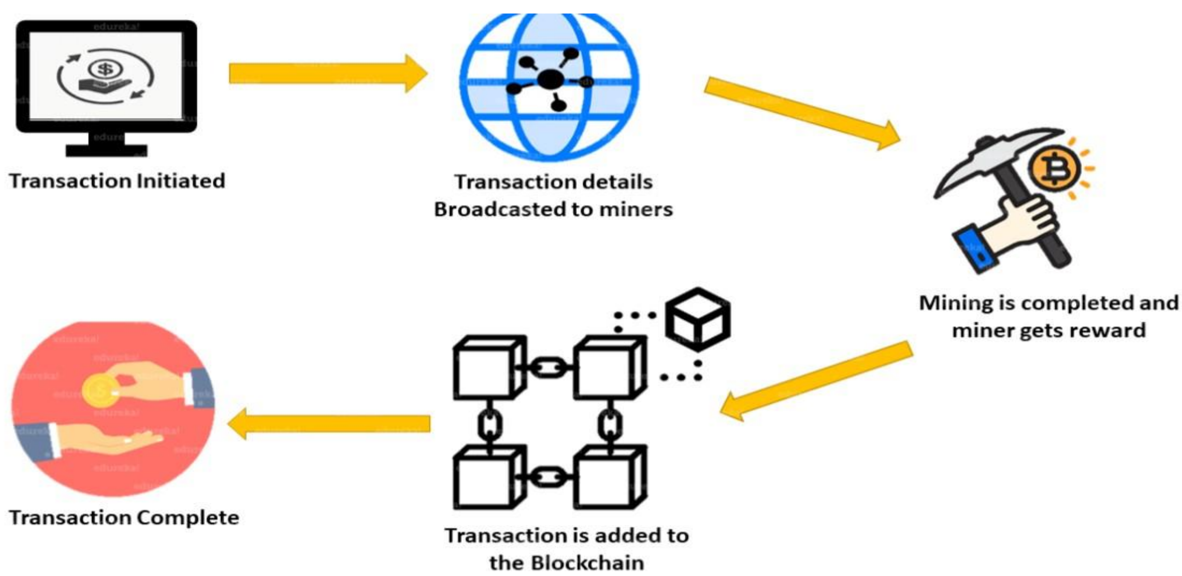
2.Process of Mining

Mining is the process of **adding new blocks** to the blockchain. It plays a crucial role in maintaining the security and integrity of the blockchain network. Mining is mainly associated with **Proof-of-Work (PoW)**–based blockchains.

Steps in the Mining Process:

1. Transactions or data are collected and prepared for a new block.
2. The miner retrieves the **previous block** from the blockchain.
3. A cryptographic puzzle called **Proof-of-Work** is solved by repeatedly calculating hashes.
4. The miner searches for a hash that satisfies a specific condition, such as having a certain number of **leading zeros**.
5. Once a valid hash is found, the new block is considered mined.
6. The mined block is broadcast to the network for verification.
7. After verification, the block is added permanently to the blockchain.

Mining requires significant **computational power**, which makes it difficult for attackers to manipulate the blockchain. It also ensures decentralization and prevents double-spending in cryptocurrency systems.



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

The validity of blocks in a blockchain is verified to ensure that the blockchain has not been tampered with. This validation process checks both the **structural integrity** and the **security rules** of the blockchain.

Steps to Check Block Validity:

1. **Previous Hash Verification**
   Each block stores the hash of the previous block. The stored hash is compared with the actual hash of the previous block. If they do not match, the block is invalid.

2. **Proof-of-Work Verification**
   The proof value of each block is checked to ensure it satisfies the Proof-of-Work condition (e.g., the hash starts with a specific number of zeros).

3. **Sequential Order Check**
   Blocks are checked to ensure they are in the correct chronological order.

4. **Genesis Block Verification**
   The first block (genesis block) is checked to ensure it has the predefined initial values.

If all these checks pass for every block, the blockchain is considered **valid**. If any block fails validation, the entire blockchain is treated as **invalid**, ensuring security and trust in the system.

Implementation:

1. Installing flask

```
C:\Users\Soham Satpute>pip install flask
Requirement already satisfied: flask in c:\users\soham satpute\appdata\local\programs\python\python312\lib\site-packages
 (3.1.0)
Requirement already satisfied: Werkzeug>=3.1 in c:\users\soham satpute\appdata\local\programs\python\python312\lib\site-
packages (from flask) (3.1.3)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\soham satpute\appdata\local\programs\python\python312\lib\site-
packages (from flask) (3.1.4)
Requirement already satisfied: itsdangerous>=2.2 in c:\users\soham satpute\appdata\local\programs\python\python312\lib\s
ite-packages (from flask) (2.2.0)
Requirement already satisfied: click>=8.1.3 in c:\users\soham satpute\appdata\local\programs\python\python312\lib\site-p
ackages (from flask) (8.1.7)
Requirement already satisfied: blinker>=1.9 in c:\users\soham satpute\appdata\local\programs\python\python312\lib\site-p
ackages (from flask) (1.9.0)
Requirement already satisfied: colorama in c:\users\soham satpute\appdata\local\programs\python\python312\lib\site-packa
ges (from click>=8.1.3->flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\soham satpute\appdata\local\programs\python\python312\lib\sit
e-packages (from Jinja2>=3.1.2->flask) (2.1.5)
```

2.Python Code: Python blockchain.py:

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

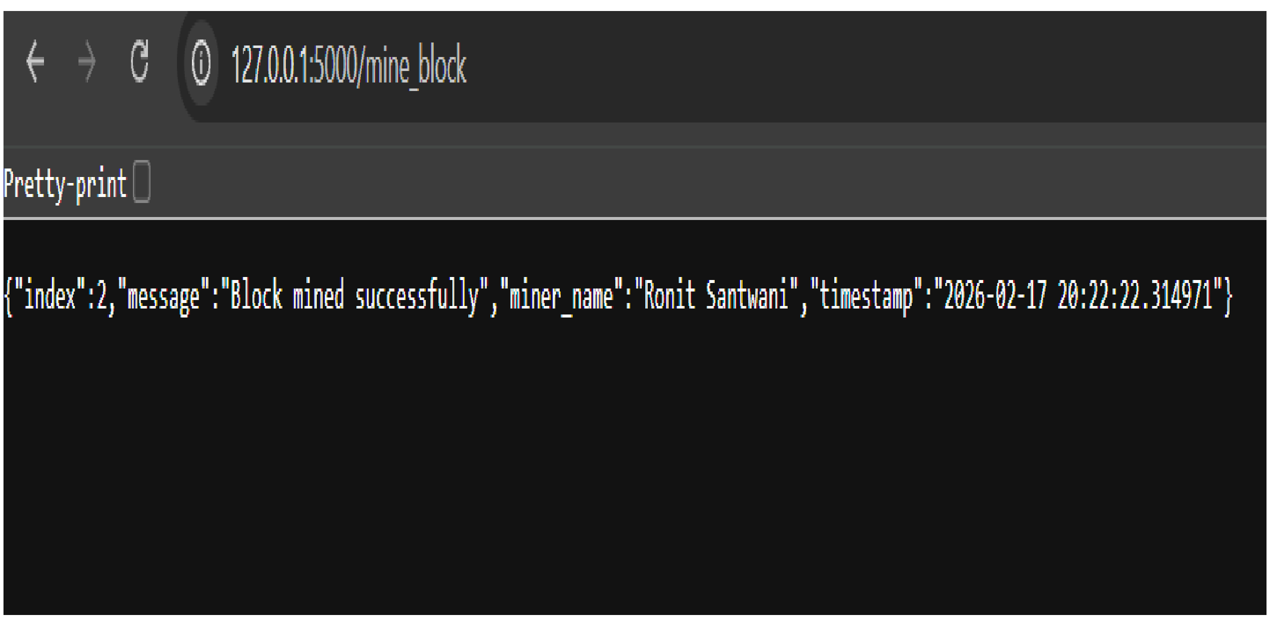
```
# Blockchain class class Blockchain:     def
__init__(self):         self.chain = []
self.create_block(proof=1,
previous_hash='0', miner_name='Genesis
Block')

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

# 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_proo
f)     previous_hash =
blockchain.hash(previous_block)

    block = blockchain.create_block(
```

```
        proof,          previous_hash,
miner_name="Soham Satpute"
    )

    response = {
        'index': block['index'],
        'message': 'Congratulations, you just mined
a block!',
        'miner_name': block['miner_name'],
        'previous_hash': block['previous_hash'],
        'proof': block['proof'],
        'timestamp': block['timestamp']
    }

    return jsonify(response), 200


# Get full blockchain
@app.route('/get_chain', methods=['GET'])
def get_chain():     response = {
'chain': blockchain.chain,
```

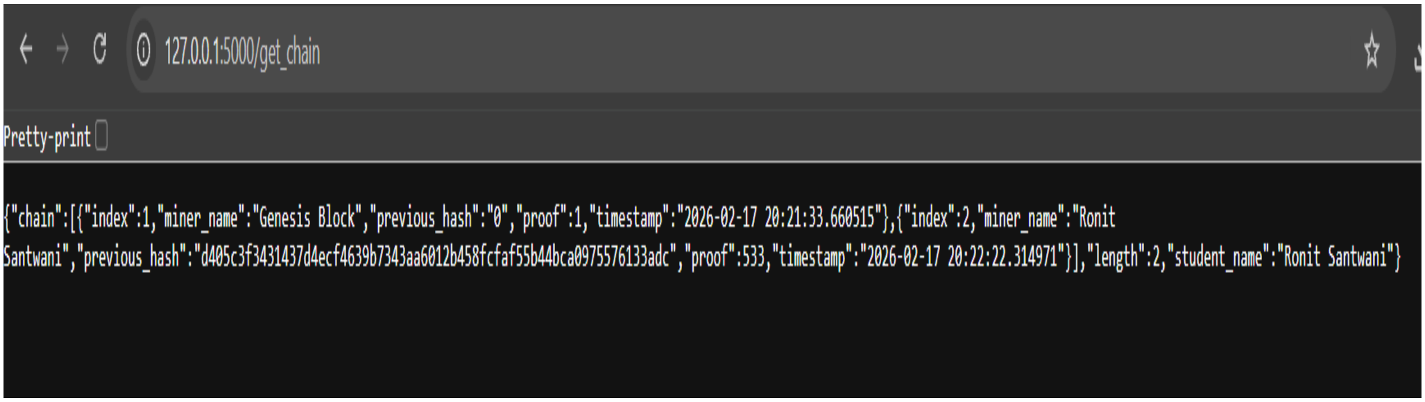
4. Output

4.1 /mine_block:

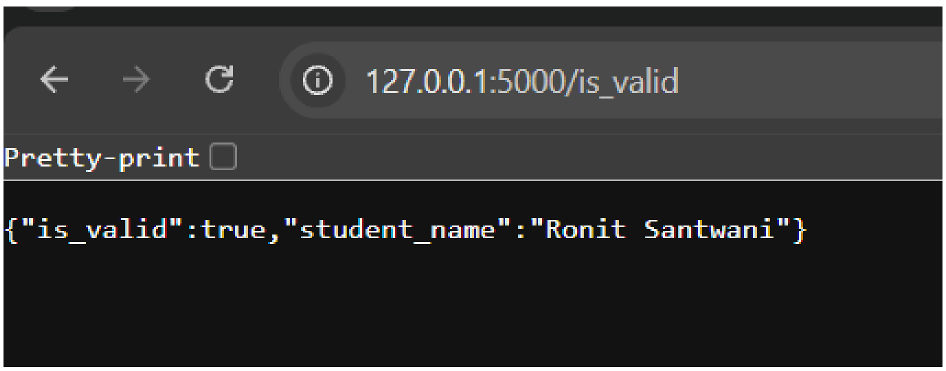


4.2 /get_chain:

```
        'length': len(blockchain.chain)
    }
    return jsonify(response), 200


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


# Run the app if
__name__ == '__main__':
    app.run(host='127.0.0.1', port=5000)
```



4.3 /is_valid:



Conclusion:

In this experiment, a basic blockchain was successfully implemented using Python. The creation of blocks, mining using a Proof of Work algorithm, and validation of the blockchain demonstrated the core principles of blockchain technology. The experiment highlighted how cryptographic hashing, consensus mechanisms, and block linking work together to ensure data integrity, security, and immutability. This implementation provides a foundational understanding of how real-world blockchain systems operate.