Sonam Chhabaidiya
D20A 16

EXPERIMENT NO 02

Aim: Create a Blockchain using Python.

Theory:

1. **What is Blockchain?**

   Blockchain is a distributed digital ledger that records transactions in a secure, transparent, and permanent way. Instead of keeping data in one central system, blockchain stores the same data across many computers in a network called nodes. The name “blockchain” comes from how information is stored — data is grouped into blocks, and each block is linked to the previous one using cryptographic hashes, which act like digital fingerprints. Because of this structure, once data is added, it becomes very difficult to change or delete, a feature known as immutability.

   Blockchain works without a central authority such as a bank or company. It follows a decentralized approach where participants in the network verify and maintain the data. Transactions are approved using a consensus mechanism, ensuring everyone agrees on the correct version of the ledger.

   Blockchain is best known for cryptocurrencies like Bitcoin and Ethereum, but it is also used in areas such as supply chain management, healthcare records, digital identity, and voting systems. In simple terms, blockchain is a secure and shared system for recording information where data is stored in connected blocks and protected using cryptography.

2. **What is Block ?**

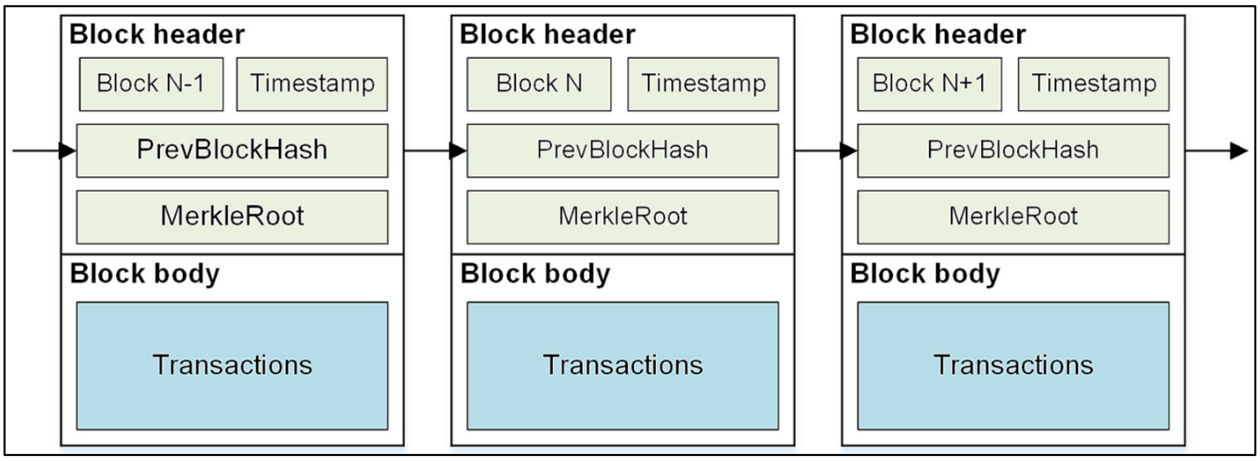
   A block is a basic unit of storage in a blockchain that holds a set of recorded transactions. It acts like a digital record book where transaction details are stored permanently. Blocks are added one after another in a specific order, creating a continuous structure called a blockchain.

   Each block is connected to the previous one, which helps maintain the correct sequence of records. Once a block is added to the chain, its data cannot be easily changed or removed. This makes blocks secure and reliable for storing information. Because of this structure, blocks play a key role in ensuring data integrity, transparency, and trust in a blockchain network.

Sonam Chhabaidiya
D20A 16

3. Components of Block

Each block in a blockchain contains several important components that help store data securely and link it to other blocks in the network.



- **Index (Block Number)**
  This represents the position of the block in the blockchain. It shows the order in which blocks are added.

- **Timestamp**
  The timestamp records the exact date and time when the block is created. It helps in tracking when transactions were added.

- **Transaction Data**
  This part contains the list of verified transactions stored inside the block.

- **Merkle Root**
  The Merkle root is a single hash value that represents all transactions in the block. It is generated using a Merkle Tree and helps quickly verify transaction integrity.

- **Previous Block Hash**
  This is the hash value of the previous block. It connects one block to another, forming the blockchain structure.

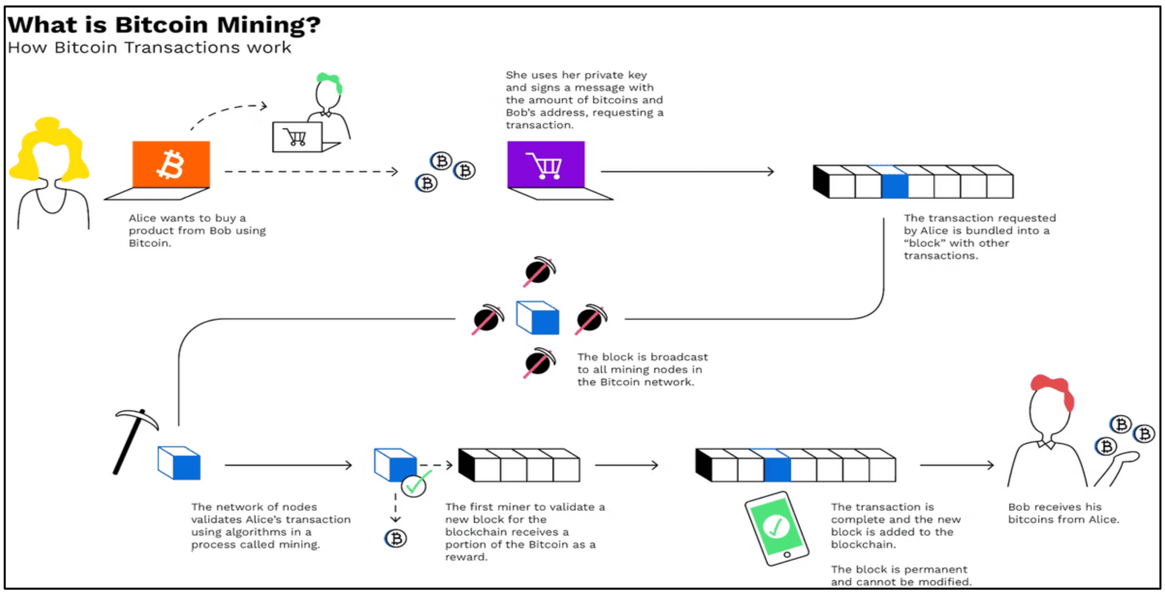
- **Nonce**
  The nonce is a special number used during mining. Miners change this value repeatedly to find a valid hash that satisfies the network's difficulty condition.

- **Hash of the Current Block**
  This is the final hash generated for the block using all the block's data. It acts as a digital fingerprint and ensures the block's security.

Together, these components make each block secure, traceable, and tamper-resistant, maintaining the integrity of the entire blockchain.

4. Process of Mining

Mining is the process by which new blocks are added to the blockchain and transactions are verified. It is mainly used in blockchains like Bitcoin that follow the Proof of Work (PoW) consensus mechanism. Mining helps maintain security, trust, and decentralization in the network


What is Bitcoin Mining?
How Bitcoin Transactions work
Alice wants to buy a product from Bob using Bitcoin.
She uses her private key and signs a message with the amount of bitcoins and Bob's address, requesting a transaction.
The transaction requested by Alice is bundled into a "block" with other transactions.
The block is broadcast to all mining nodes in the Bitcoin network.
The network of nodes validates Alice's transaction using algorithms in a process called mining.
The first miner to validate a new block for the blockchain receives a portion of the Bitcoin as a reward.
The transaction is complete and the new block is added to the blockchain.
The block is permanent and cannot be modified.
Bob receives his bitcoins from Alice.

Step 1: Collection of Transactions
When users make transactions, they are first broadcast to the blockchain network. These transactions are stored temporarily in a pool called the **memorypool**. Miners select valid transactions from this pool to include in a new block.

Step 2: Creating the Block
The miner groups selected transactions into a block. The block also includes important details like the previous block hash, Merkle root, timestamp, and nonce inside the block header.

Step 3: Solving the Cryptographic Puzzle
Mining involves solving a difficult mathematical problem using a cryptographic hash function (like SHA-256 in Bitcoin). Miners repeatedly change the nonce value and generate a new hash for the block header.

The goal is to find a hash that is less than a target value set by the network. This is called the **Proof of Work**.

Step 4: Finding the Valid Hash
This process requires a lot of trial and error and computational power. When a miner finally finds a hash that meets the difficulty requirement, the block is considered successfully mined.

Step 5: Block Broadcasting and Verification
The mined block is broadcast to the network. Other nodes verify:

- Transactions are valid
- Hash is correct
- Block follows blockchain rules

If everything is correct, the block is added to the blockchain.

Step 6: Mining Reward
The miner who successfully mines the block receives a block reward along with transaction fees. This reward acts as an incentive for miners to continue securing the network.

5. How to check the validity of blocks in Blockchain?

Checking the validity of a block is an essential step to ensure that the blockchain remains **secure, accurate, and tamper-proof**. Every block added to the blockchain must be verified before it is accepted by the network.

Steps to Validate a Block:

1. **Verify the Previous Block Hash**
   Each block contains the hash of the previous block. Nodes in the network check that this hash matches the hash of the actual previous block. If it doesn't match, the block is considered invalid. This ensures the chain is continuous and unbroken.

2. **Validate Transactions**
   All transactions inside the block are verified to ensure they are legitimate. This includes checking that the sender has sufficient balance, digital signatures are correct, and transactions follow network rules.

3. **Check the Block Hash**
   The node recalculates the hash of the block using the block's data and the nonce. If the recalculated hash matches the hash stored in the block and satisfies the network's difficulty requirement, the block is valid.

Sonam Chhabaidiya
D20A 16

4. **Verify the Merkle Root**
   The Merkle root in the block header is checked to ensure that all transactions are intact and have not been tampered with. If even a single transaction is altered, the Merkle root will not match.

5. **Consensus Verification**
   Finally, the network uses a **consensus mechanism** (like Proof of Work or Proof of Stake) to agree that the block is valid. Only blocks that are approved by the majority of nodes are added to the blockchain.

Code:

Implementation of a Simple Blockchain with Proof of Work and Flask API

Blockchain.py

```
import datetime
import
hashlib
import json

class Blockchain:

   def _init_(self):
      self.chain = []
      self.transactions = []       #
Transaction list
      self.difficulty = 3          #
"000" leading zeros
      self.create_genesis_block(
   ) def
   create_genesis_block(self):
      genesis_block = {
         'index': 1,
         'timestamp':
str(datetime.datetime.now()),
         'transactions': [],
         'nonce': 0,
         'previous_hash': '0'
      }
      genesis_block['hash'] =
self.calculate_hash(genesis_block)
      self.chain.append(genesis_block)

   def create_Transactions(self,
sender, receiver, amount):
      transaction = {
         'sender': sender,
         'receiver': receiver,
         'amount': amount
```

```
}self.transactions.append(transaction)
   def calculate_hash(self, block):
      block_copy = block.copy()
      block_copy.pop('hash', None)
      encoded_block =
json.dumps(block_copy,
sort_keys=True).encode()
      return
hashlib.sha256(encoded_block).hexdigest
(
)
   def proof_of_work(self, block):
      nonce = 0
      while True:
         block['nonce'] =
         nonce hash_value =
self.calculate_hash(block)
         if hash_value.startswith('0' *
self.difficulty):
            return nonce, hash_value
         nonce += 1
   def mine_block(self):
      if len(self.transactions) == 0:
         return None  # Mine only if
transactions exist
      previous_block = self.chain[-
      1] block = {
         'index': len(self.chain) + 1,
         'timestamp':
str(datetime.datetime.now()),
         'transactions':
         self.transactions, 'nonce': 0,
         'previous_hash':
```

```
        nonce, hash_value =
self.proof_of_work(block)

        block['nonce'] = nonce
        block['hash'] = hash_value

        # Clear transaction pool after
mining
        self.transactions = []

        # Add mined block to chain
        self.chain.append(block)

        return block

    # Validate Blockchain

    def is_chain_valid(self):

        for i in range(1, len(self.chain)):

            current = self.chain[i]
            previous = self.chain[i - 1]

            # Check previous hash linkage
            if current['previous_hash'] !=
previous['hash']:
                return False

            # Recalculate hash
            recalculated_hash =
self.calculate_hash(current)

            if recalculated_hash !=
current['hash']:
                return False

            # Check Proof of Work
condition
            if not
current['hash'].startswith('0' *
self.difficulty):
                return False

        return True
```

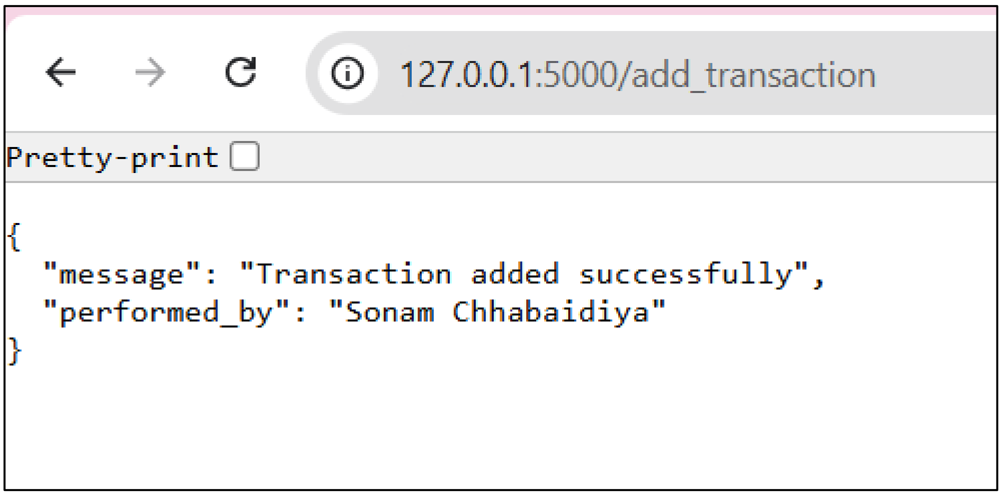
Sonam Chhabaidiya
D20A 16

main.py

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

app = Flask(_name_)
blockchain = Blockchain()

@app.route('/add_transaction',
methods=['GET'])
def add_transaction():
    blockchain.create_Transactions("Alic
    e",
"Bob", 50)
    return jsonify({'message':
'Transaction added successfully'}), 200
@app.route('/mine_block',
methods=['GET'])
def mine_block():
    block =
    blockchain.mine_block() if
    block is None:
        return jsonify({'message': 'No
transactions to mine'}), 400
    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(" Blockchain running with
3 leading zero difficulty")
    app.run(debug=True)
```

Output :

➢ Adding Transaction to Transaction Pool

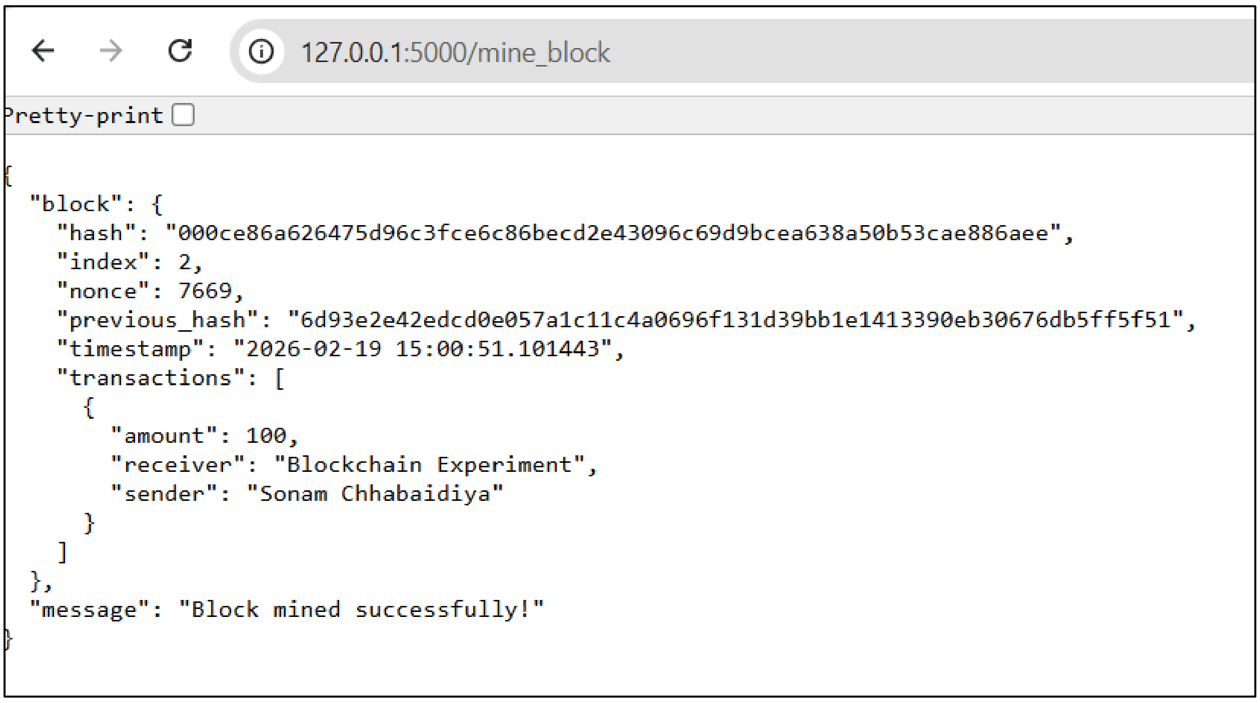

127.0.0.1:5000/add_transaction

Pretty-print ☐

```
{
  "message": "Transaction added successfully",
  "performed_by": "Sonam Chhabaidiya"
}
```

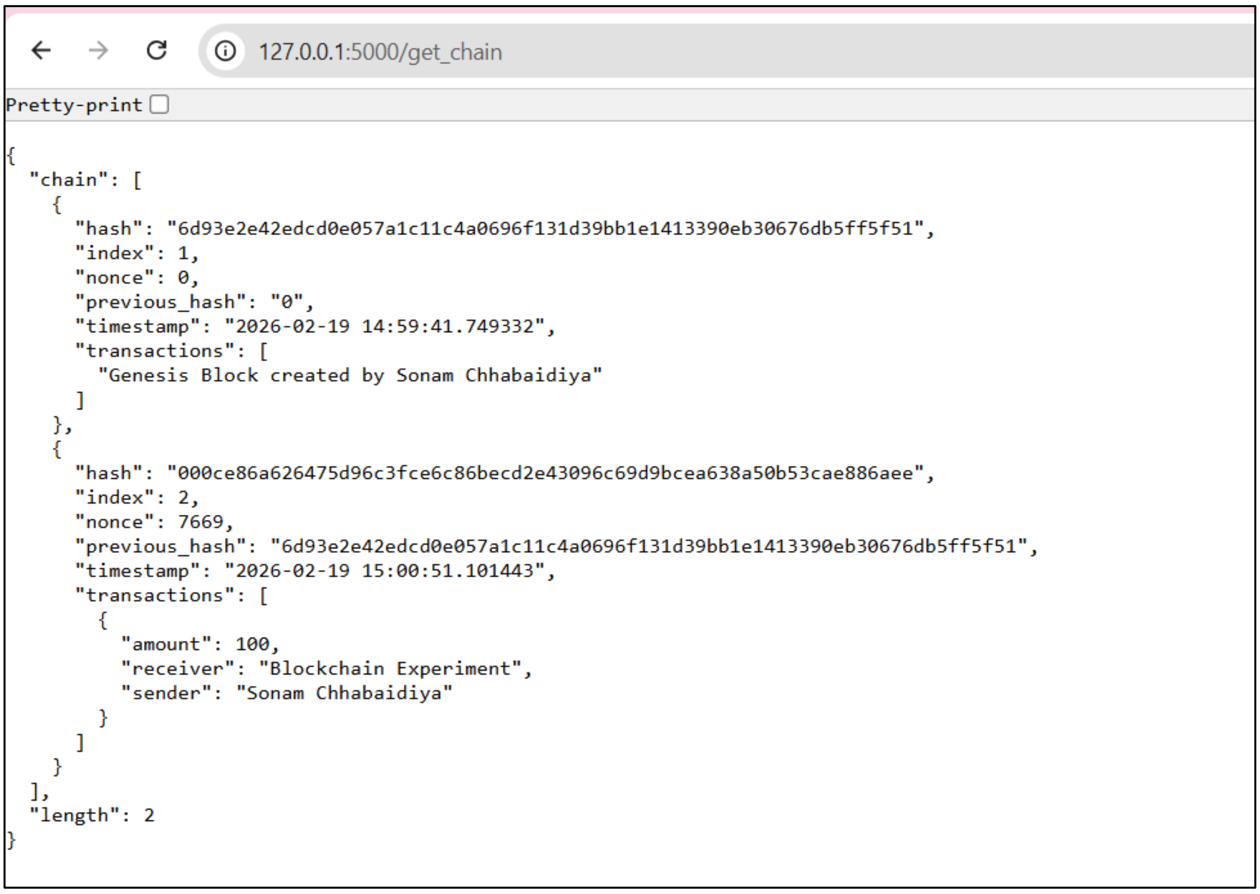
Sonam Chhabaidiya
D20A 16

➤ Mining a New Block Using Proof of Work



```
{
  "block": {
    "hash": "000ce86a626475d96c3fce6c86becd2e43096c69d9bcea638a50b53cae886aee",
    "index": 2,
    "nonce": 7669,
    "previous_hash": "6d93e2e42edcd0e057a1c11c4a0696f131d39bb1e1413390eb30676db5ff5f51",
    "timestamp": "2026-02-19 15:00:51.101443",
    "transactions": [
      {
        "amount": 100,
        "receiver": "Blockchain Experiment",
        "sender": "Sonam Chhabaidiya"
      }
    ]
  },
  "message": "Block mined successfully!"
}
```

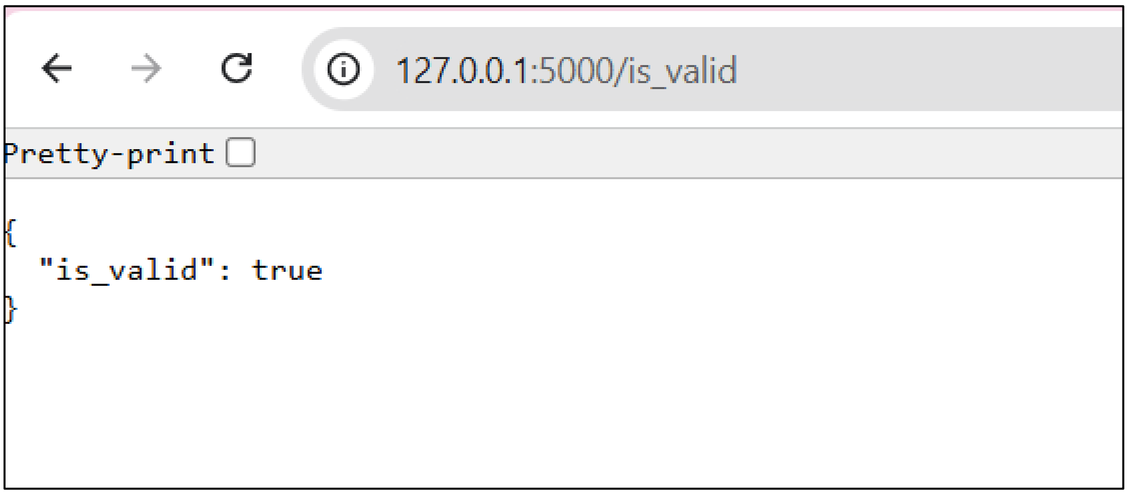
➤ Displaying the Blockchain



```
{
  "chain": [
    {
      "hash": "6d93e2e42edcd0e057a1c11c4a0696f131d39bb1e1413390eb30676db5ff5f51",
      "index": 1,
      "nonce": 0,
      "previous_hash": "0",
      "timestamp": "2026-02-19 14:59:41.749332",
      "transactions": [
        "Genesis Block created by Sonam Chhabaidiya"
      ]
    },
    {
      "hash": "000ce86a626475d96c3fce6c86becd2e43096c69d9bcea638a50b53cae886aee",
      "index": 2,
      "nonce": 7669,
      "previous_hash": "6d93e2e42edcd0e057a1c11c4a0696f131d39bb1e1413390eb30676db5ff5f51",
      "timestamp": "2026-02-19 15:00:51.101443",
      "transactions": [
        {
          "amount": 100,
          "receiver": "Blockchain Experiment",
          "sender": "Sonam Chhabaidiya"
        }
      ]
    }
  ],
  "length": 2
}
```

Sonam Chhabaidiya
D20A 16

➢ Blockchain Validation



127.0.0.1:5000/is_valid

Pretty-print

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

Conclusion :

A blockchain system was successfully implemented using Python and Flask. Transactions were added to a transaction pool and blocks were mined only when transactions were available. Proof of Work was used to find a golden nonce such that the block hash starts with three leading zeros. Each block was securely linked using cryptographic hashing. The blockchain validation confirmed data integrity and immutability.