Ronak Katariya
D20A / 29

EXPERIMENT NO: 03

AIM: Create a Cryptocurrency using Python and perform mining in the Blockchain created.

THEORY:

1. Introduction to Blockchain

Blockchain is a distributed ledger system that records transactions securely across a network of multiple computers, known as nodes. Instead of relying on a single central authority, every participating node maintains a synchronized copy of the ledger, ensuring transparency and reliability.

A blockchain is composed of blocks, and each block contains:

- A list of verified transactions
- A timestamp indicating when the block was created
- The hash value of the previous block
- Its own unique cryptographic hash

These blocks are linked together using cryptographic hashes, forming a secure chain. If any information inside a block is altered, its hash changes, which breaks the connection with the subsequent block. This structure makes tampering easily detectable and guarantees data integrity, security, and immutability.

2. Mining Mechanism

Mining is the process responsible for validating transactions and adding new blocks to the blockchain. The mining procedure generally involves:

- Gathering unconfirmed transactions
- Constructing a new block
- Solving a complex mathematical puzzle using the Proof-of-Work (PoW) algorithm

Under the Proof-of-Work system, miners repeatedly adjust a nonce value in the block header to generate a hash that meets a specified difficulty target, such as producing a hash with a certain number of leading zeros. When a miner successfully finds a valid hash, the block is shared with the network for verification and then appended to the blockchain. In return, the miner receives a cryptocurrency reward for contributing computational power.

3. Multi-Node Blockchain Configuration

In this practical setup, a decentralized blockchain network was created using three separate nodes operating on ports 5001, 5002, and 5003.

Each node:

- Operates autonomously
- Maintains its own blockchain copy
- Communicates and synchronizes with other nodes

This arrangement illustrates how blockchain networks function without a centralized control system, while still maintaining consistency and synchronization across all participants.

4. Consensus Protocol

To ensure consistency throughout the network, a consensus algorithm known as the Longest Valid Chain Rule is implemented.

When different versions of the blockchain exist, each node evaluates them and adopts the longest chain that satisfies validation rules. This mechanism resolves temporary conflicts and guarantees that all nodes eventually agree on a unified and accurate transaction history.

5. Transactions and Mining Rewards

In a blockchain-based cryptocurrency system, each transaction includes:

- The sender's address
- The receiver's address
- The transfer amount

When miners create a new block, they include all pending transactions in that block. Additionally, a special reward transaction is generated to compensate the miner with newly issued cryptocurrency. This reward system motivates participants to continue supporting and securing the blockchain network.

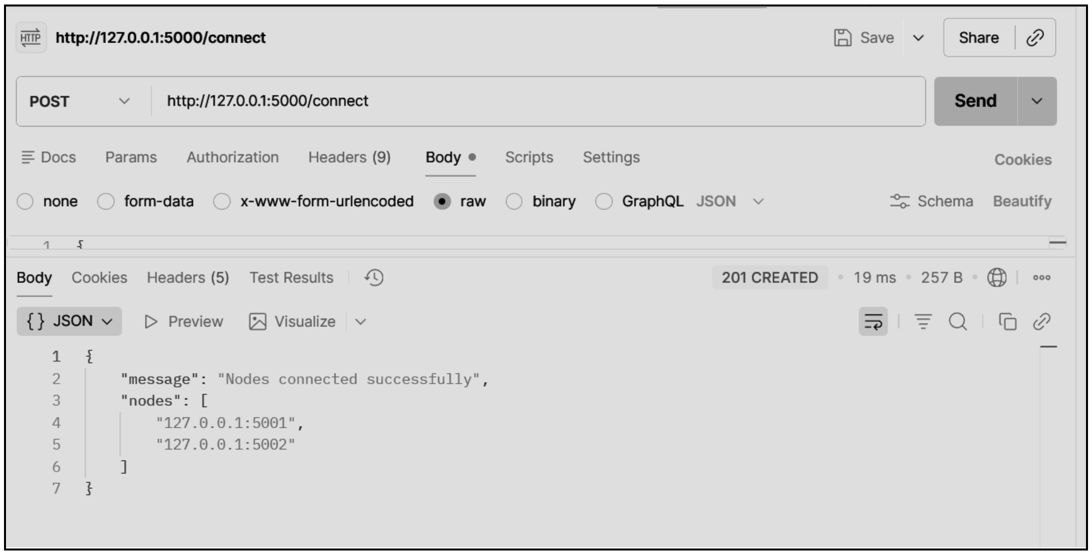
CODE:

```
import datetime
import hashlib
import json
import requests
from flask import Flask, jsonify, request
from uuid import uuid4
from urllib.parse import urlparse


class SimpleBlockchain:
    def __init__(self):
        self.ledger = []
        self.pending_txns = []
        self.network_nodes = set()
        self._create_genesis_block()

    def _create_genesis_block(self):
        self.create_block(proof=1, prev_hash="0")

    def create_block(self, proof, prev_hash):
        block = {
            "block_no": len(self.ledger) + 1,
            "time": str(datetime.datetime.now()),
            "proof": proof,
            "prev_hash": prev_hash,
            "transactions": self.pending_txns
        }
        self.pending_txns = []
        self.ledger.append(block)
        return block

    def last_block(self):
        return self.ledger[-1]

    def compute_proof(self, last_proof):
        new_proof = 1
        while True:
            guess = hashlib.sha256(
                str(new_proof**2 - last_proof**2).encode()
            ).hexdigest()
            if guess[:4] == "0000":
                return new_proof
            new_proof += 1

    def generate_hash(self, block):
        encoded = json.dumps(block, sort_keys=True).encode()
        return hashlib.sha256(encoded).hexdigest()

    def validate_chain(self, chain):
        prev_block = chain[0]
        idx = 1

        while idx < len(chain):
            curr_block = chain[idx]

            if curr_block["prev_hash"] != self.generate_hash(prev_block):
```

```
            return False

        prev_proof = prev_block["proof"]
        curr_proof = curr_block["proof"]

        check_hash = hashlib.sha256(
            str(curr_proof**2 - prev_proof**2).encode()
        ).hexdigest()

        if check_hash[:4] != "0000":
            return False

        prev_block = curr_block
        idx += 1

    return True

def add_txn(self, sender, receiver, amount):
    self.pending_txns.append({
        "from": sender,
        "to": receiver,
        "value": amount
    })
    return self.last_block()["block_no"] + 1

def register_node(self, node_url):
    parsed = urlparse(node_url)
    self.network_nodes.add(parsed.netloc)

def sync_chain(self):
    longest = None
    max_len = len(self.ledger)

    for node in self.network_nodes:
        response = requests.get(f"http://{node}/get_chain")
        if response.status_code == 200:
            length = response.json()["length"]
            chain = response.json()["chain"]

            if length > max_len and self.validate_chain(chain):
                max_len = length
                longest = chain

    if longest:
            self.ledger = longest
            return True
        return False


app = Flask(__name__)
node_id = str(uuid4()).replace("-", "")
blockchain = SimpleBlockchain()


@app.route("/mine", methods=["GET"])
def mine():
    last_block = blockchain.last_block()
    proof = blockchain.compute_proof(last_block["proof"])
    prev_hash = blockchain.generate_hash(last_block)

    blockchain.add_txn(
        sender=node_id,
        receiver="Shravani",
        amount=1
    )

    block = blockchain.create_block(proof, prev_hash)

    return jsonify({
        "status": "Block mined successfully!",
        "block": block
    }), 200


@app.route("/transaction", methods=["POST"])
def create_transaction():
    data = request.get_json()
    required = ["sender", "receiver", "amount"]

    if not all(k in data for k in required):
        return "Invalid transaction data", 400

    block_no = blockchain.add_txn(
        data["sender"],
        data["receiver"],
        data["amount"]
    )
    return jsonify({
        "message": f"Transaction will be added to block {block_no}"
    }), 201


@app.route("/connect", methods=["POST"])
def connect_nodes():
    data = request.get_json()
    nodes = data.get("nodes")

    if not nodes:
        return "No nodes provided", 400

    for node in nodes:
        blockchain.register_node(node)

    return jsonify({
        "message": "Nodes connected successfully",
        "nodes": list(blockchain.network_nodes)
    }), 201


@app.route("/sync", methods=["GET"])
def sync():
    replaced = blockchain.sync_chain()

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

app.run(host="0.0.0.0")
```

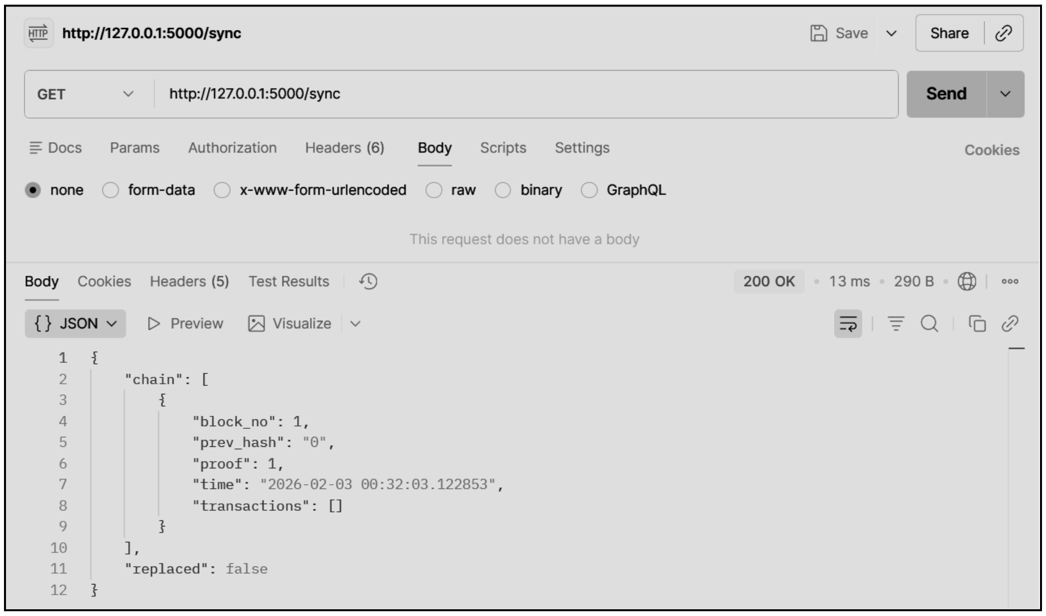
OUTPUT:

Connect the blocks:


POST http://127.0.0.1:5000/connect

201 CREATED • 19 ms • 257 B

```
{
    "message": "Nodes connected successfully",
    "nodes": [
        "127.0.0.1:5001",
        "127.0.0.1:5002"
    ]
}
```

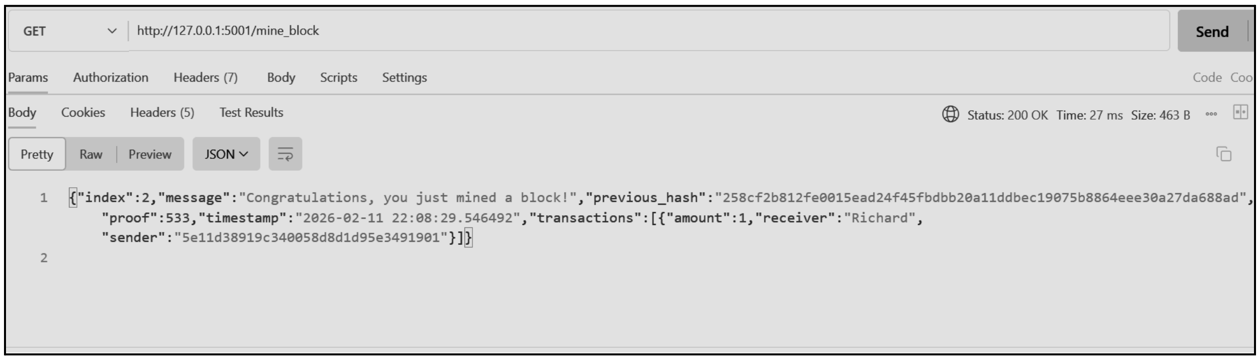

GET http://127.0.0.1:5000/sync

This request does not have a body

200 OK • 13 ms • 290 B

```
{
    "chain": [
        {
            "block_no": 1,
            "prev_hash": "0",
            "proof": 1,
            "time": "2026-02-03 00:32:03.122853",
            "transactions": []
        }
    ],
    "replaced": false
}
```

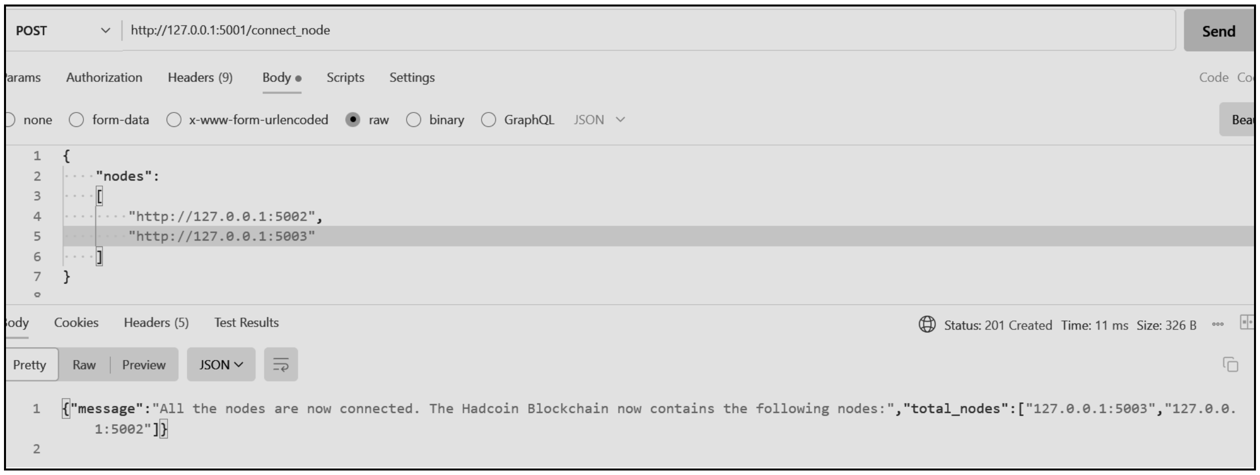
Mined a block:


GET http://127.0.0.1:5001/mine_block

Status: 200 OK Time: 27 ms Size: 463 B

```
{"index":2,"message":"Congratulations, you just mined a block!","previous_hash":"258cf2b812fe0015ead24f45fbdbb20a11ddbec19075b8864eee30a27da688ad",
    "proof":533,"timestamp":"2026-02-11 22:08:29.546492","transactions":[{"amount":1,"receiver":"Richard",
    "sender":"5e11d38919c340058d8d1d95e3491901"}]}
```

Connected the nodes:

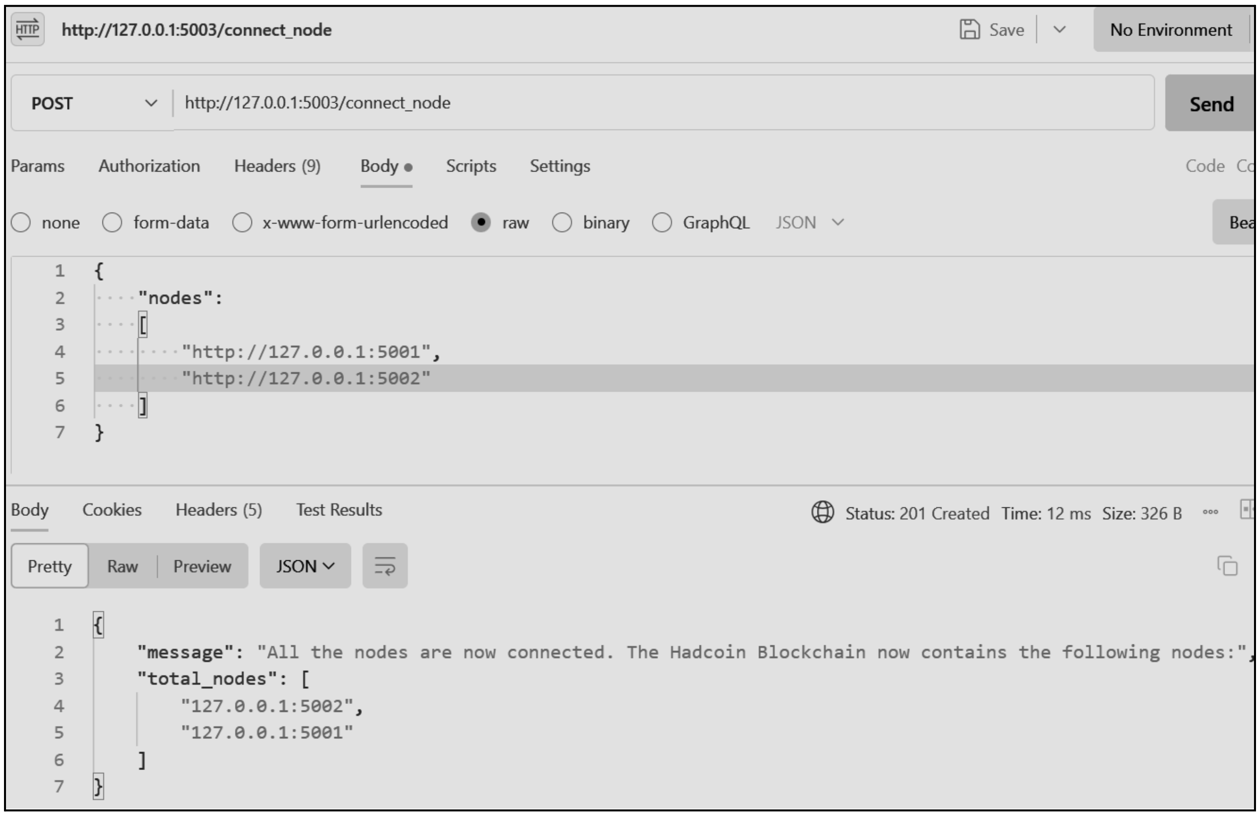


http://127.0.0.1:5001/connect_node

```
{
    "nodes":
    [
        "http://127.0.0.1:5002",
        "http://127.0.0.1:5003"
    ]
}
```

Status: 201 Created Time: 11 ms Size: 326 B

```
{"message":"All the nodes are now connected. The Hadcoin Blockchain now contains the following nodes:","total_nodes":["127.0.0.1:5003","127.0.0.
    1:5002"]}
```



http://127.0.0.1:5003/connect_node

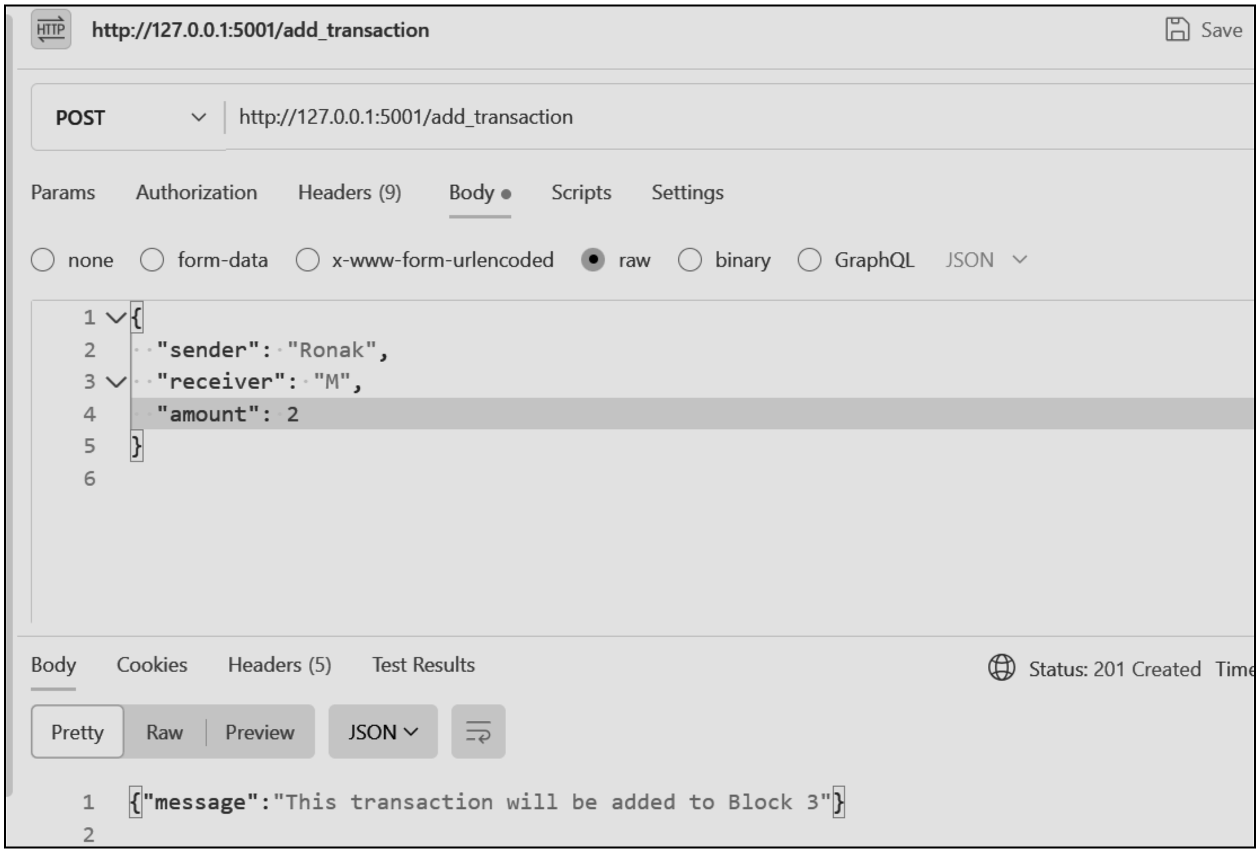
http://127.0.0.1:5003/connect_node

```
{
    "nodes":
    [
        "http://127.0.0.1:5001",
        "http://127.0.0.1:5002"
    ]
}
```

Status: 201 Created Time: 12 ms Size: 326 B

```
{
    "message": "All the nodes are now connected. The Hadcoin Blockchain now contains the following nodes:",
    "total_nodes": [
        "127.0.0.1:5002",
        "127.0.0.1:5001"
    ]
}
```

Transaction added:


http://127.0.0.1:5001/add_transaction

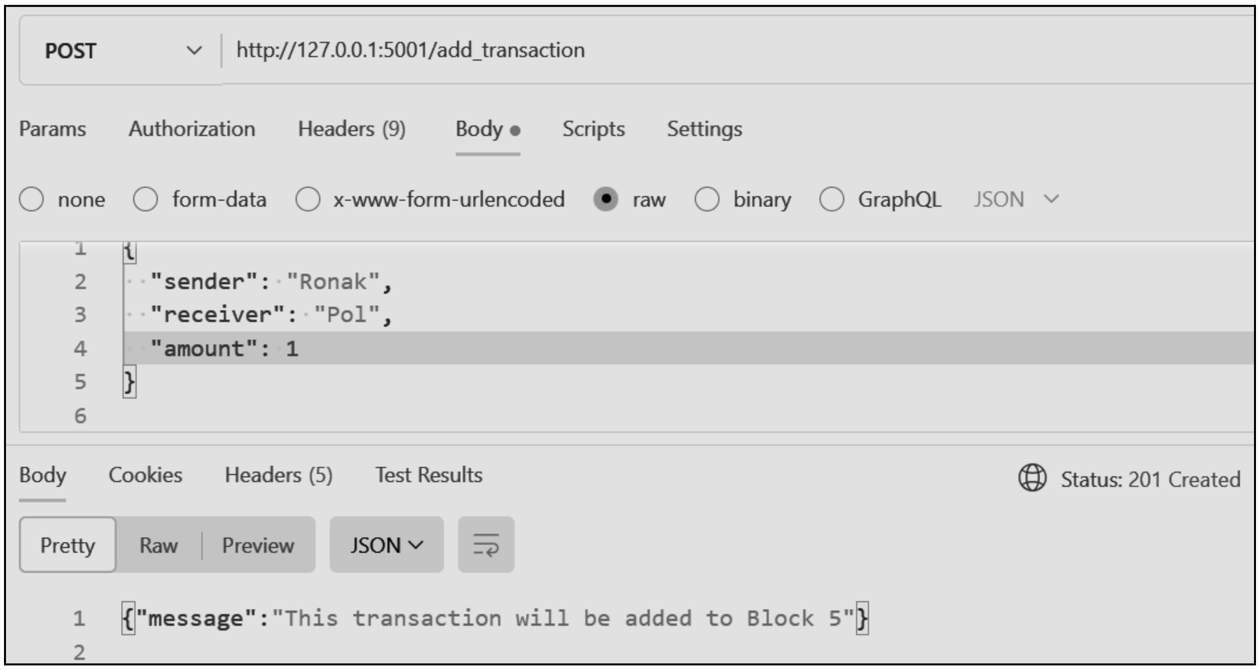
POST http://127.0.0.1:5001/add_transaction

Body: raw JSON

```
{
  "sender": "Ronak",
  "receiver": "M",
  "amount": 2
}
```

Status: 201 Created

```
{"message":"This transaction will be added to Block 3"}
```

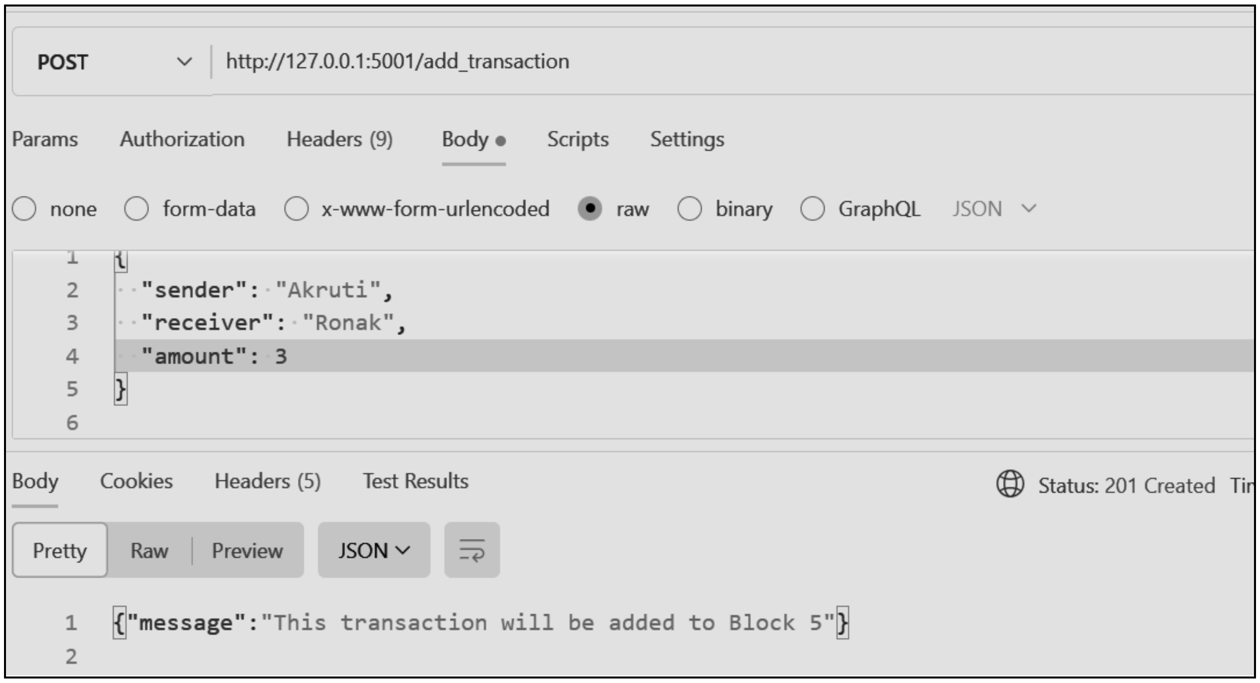

POST http://127.0.0.1:5001/add_transaction

Body: raw JSON

```
{
  "sender": "Ronak",
  "receiver": "Pol",
  "amount": 1
}
```

Status: 201 Created

```
{"message":"This transaction will be added to Block 5"}
```



POST http://127.0.0.1:5001/add_transaction

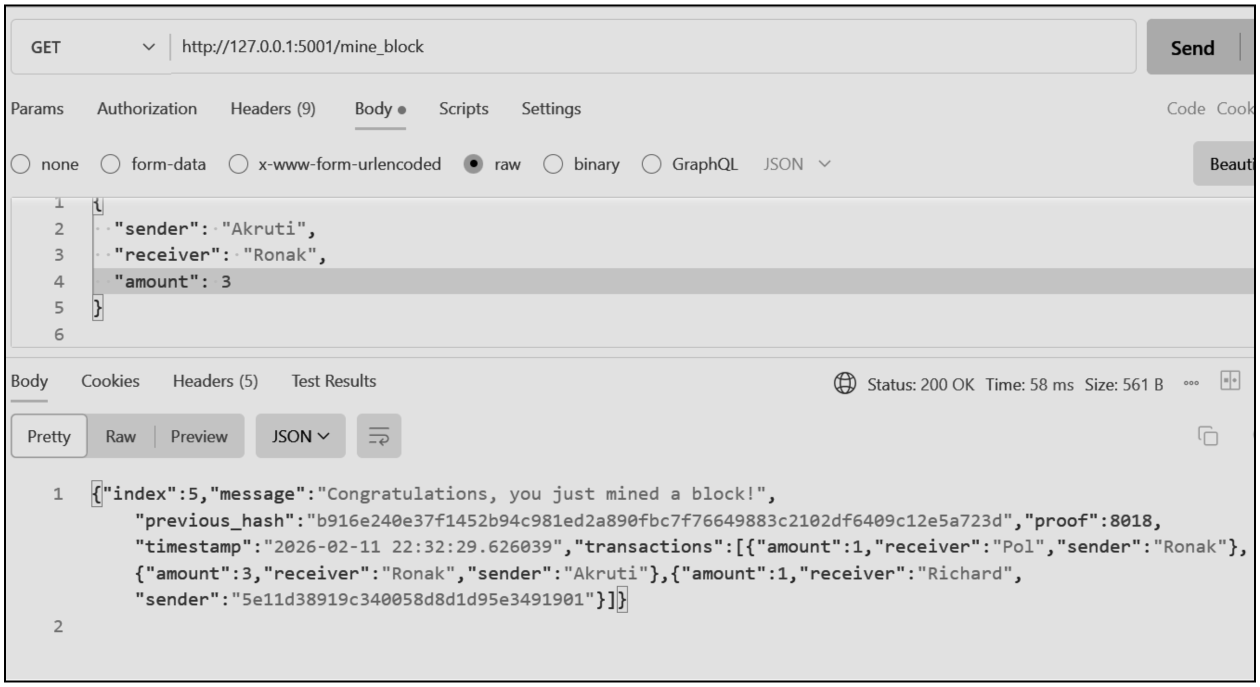
Body: raw, JSON

```
{
  "sender": "Akruti",
  "receiver": "Ronak",
  "amount": 3
}
```

Status: 201 Created

```
{"message":"This transaction will be added to Block 5"}
```

Get transaction:



GET http://127.0.0.1:5001/mine_block

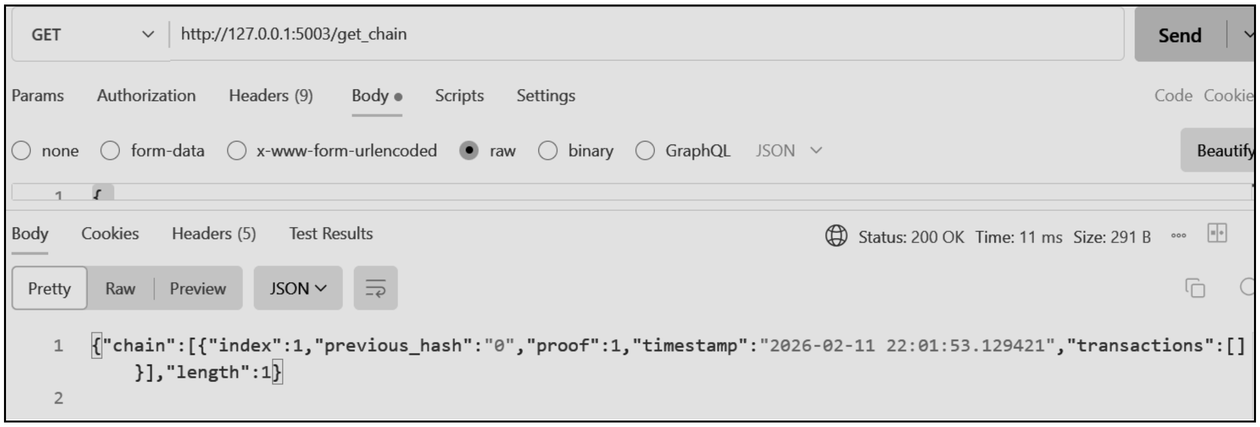
Body: raw, JSON

```
{
  "sender": "Akruti",
  "receiver": "Ronak",
  "amount": 3
}
```

Status: 200 OK Time: 58 ms Size: 561 B

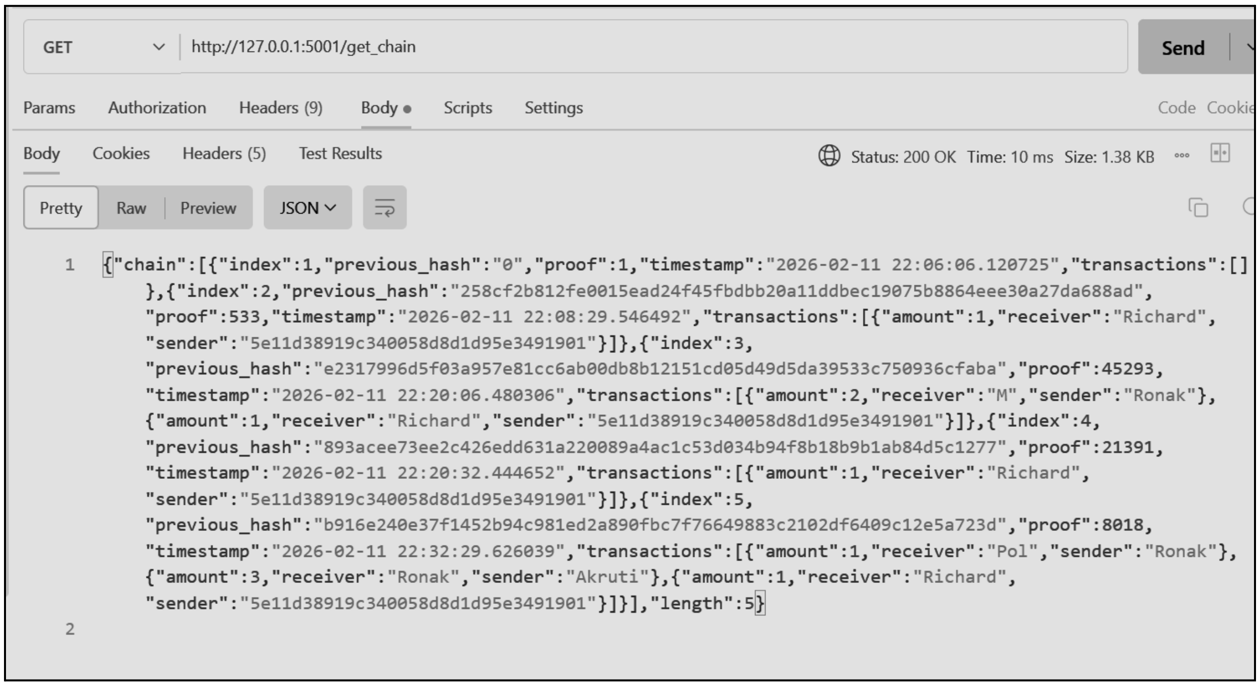
```
{"index":5,"message":"Congratulations, you just mined a block!",
    "previous_hash":"b916e240e37f1452b94c981ed2a890fbc7f76649883c2102df6409c12e5a723d","proof":8018,
    "timestamp":"2026-02-11 22:32:29.626039","transactions":[{"amount":1,"receiver":"Pol","sender":"Ronak"},
    {"amount":3,"receiver":"Ronak","sender":"Akruti"},{"amount":1,"receiver":"Richard",
    "sender":"5e11d38919c340058d8d1d95e3491901"}]}
```

Get chain:


GET http://127.0.0.1:5003/get_chain

Status: 200 OK Time: 11 ms Size: 291 B

```
{"chain":[{"index":1,"previous_hash":"0","proof":1,"timestamp":"2026-02-11 22:01:53.129421","transactions":[]
    }],"length":1}
```


GET http://127.0.0.1:5001/get_chain

Status: 200 OK Time: 10 ms Size: 1.38 KB

```
{"chain":[{"index":1,"previous_hash":"0","proof":1,"timestamp":"2026-02-11 22:06:06.120725","transactions":[]
    },{"index":2,"previous_hash":"258cf2b812fe0015ead24f45fbdbb20a11ddbec19075b8864eee30a27da688ad",
    "proof":533,"timestamp":"2026-02-11 22:08:29.546492","transactions":[{"amount":1,"receiver":"Richard",
    "sender":"5e11d38919c340058d8d1d95e3491901"}]},{"index":3,
    "previous_hash":"e2317996d5f03a957e81cc6ab00db8b12151cd05d49d5da39533c750936cfaba","proof":45293,
    "timestamp":"2026-02-11 22:20:06.480306","transactions":[{"amount":2,"receiver":"M","sender":"Ronak"},
    {"amount":1,"receiver":"Richard","sender":"5e11d38919c340058d8d1d95e3491901"}]},{"index":4,
    "previous_hash":"893acee73ee2c426edd631a220089a4ac1c53d034b94f8b18b9b1ab84d5c1277","proof":21391,
    "timestamp":"2026-02-11 22:20:32.444652","transactions":[{"amount":1,"receiver":"Richard",
    "sender":"5e11d38919c340058d8d1d95e3491901"}]},{"index":5,
    "previous_hash":"b916e240e37f1452b94c981ed2a890fbc7f76649883c2102df6409c12e5a723d","proof":8018,
    "timestamp":"2026-02-11 22:32:29.626039","transactions":[{"amount":1,"receiver":"Pol","sender":"Ronak"},
    {"amount":3,"receiver":"Ronak","sender":"Akruti"},{"amount":1,"receiver":"Richard",
    "sender":"5e11d38919c340058d8d1d95e3491901"}]}],"length":5}
```

Conclusion

In this experiment, we developed a basic blockchain network across two devices connected within the same local network. Each device hosted its own blockchain node on port 5000, providing functionalities to add transactions, mine new blocks, and replace the existing chain when necessary. By sending cURL requests, we verified the system's functionality by adding transactions, mining them into blocks, and synchronizing both nodes using the chain replacement feature.

Through this implementation, we gained practical insight into how blockchain achieves decentralization, immutability, and consensus among distributed nodes. Although the model was simplified compared to real-world blockchain platforms, it effectively demonstrated the essential concepts of distributed ledger technology, node communication, and maintaining a consistent and secure transaction record across a network.