Shreya Sawant
D20A - 55

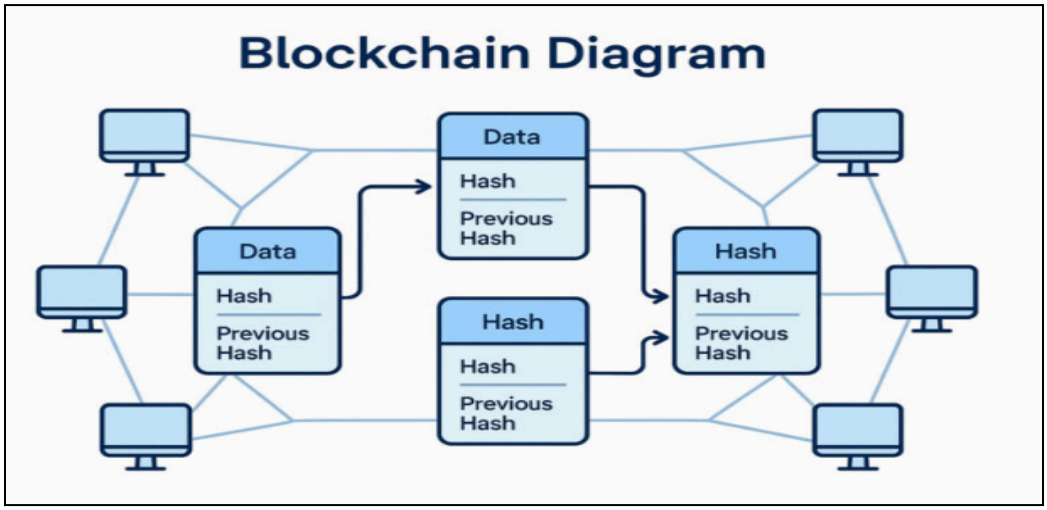
Experiment. 3

Aim - Create a Cryptocurrency using Python and perform mining in the Blockchain created.

Theory -

1. Blockchain Overview

Blockchain is a distributed and decentralized digital ledger that records data in the form of interconnected blocks. Unlike traditional centralized databases, which are controlled by a single authority (such as a bank or organization), blockchain operates on a peer-to-peer network where multiple participants maintain copies of the same ledger. This decentralized structure improves transparency, security, and reliability of data storage.



Each block in a blockchain contains transaction data, a timestamp indicating when the block was created, the hash of the previous block, and its own unique cryptographic hash. The previous block's hash securely links blocks together, forming an immutable chain. If any data within a block is altered, its hash changes, breaking the connection with subsequent blocks. To restore the chain, all following blocks would need to be recalculated, which is practically infeasible in a large network.

Each block contains:

- Transaction details
- Timestamp (time of creation)
- Hash of the previous block
- Its own unique hash

2. Mining

Mining is a fundamental process in blockchain that validates transactions and adds new blocks to the network. It begins with the collection of pending transactions from the transaction pool, which are then grouped into a candidate block. Miners compete to solve a cryptographic puzzle known as Proof-of-Work by finding a suitable nonce that produces a hash meeting the network's difficulty criteria, typically requiring a specific number of leading zeros.

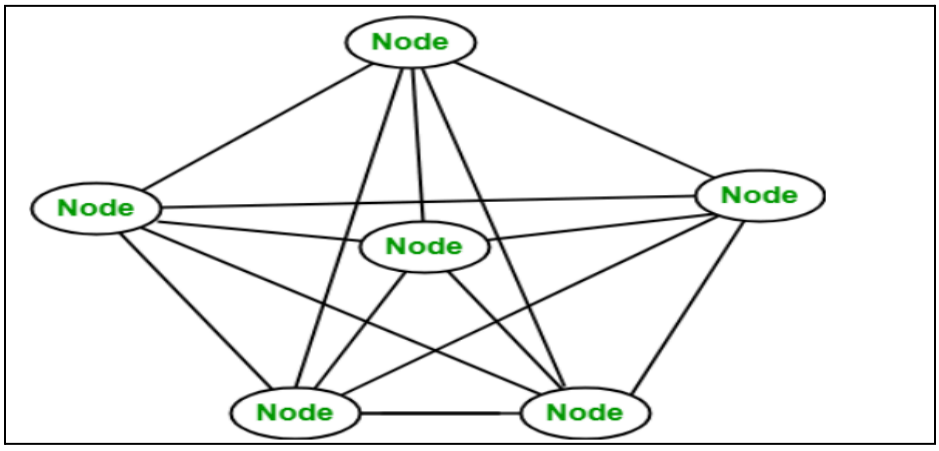
Once a miner successfully finds a valid hash, the new block is broadcast to all connected peers in the network. Other nodes verify the block before accepting it into their copy of the blockchain. Upon successful validation, the block becomes part of the permanent ledger. As an incentive for their computational effort, miners receive a cryptocurrency reward along with transaction fees.

Steps in mining:

1. Collect pending transactions
2. Try different nonce values to generate a valid hash
3. Once a valid hash is found, the block is created
4. The block is shared with all other nodes
5. If verified, it is added to the blockchain
6. Miner receives a reward

3. Multi-Node Blockchain Network

A blockchain operates as a distributed system composed of multiple independent nodes that communicate with each other over a peer-to-peer network. In this laboratory simulation, three separate blockchain nodes are implemented on different ports: **5001, 5002, and 5003**. Each node functions as an autonomous participant in the network, maintaining its own copy of the blockchain and validating transactions independently.



These nodes exchange information regarding newly mined blocks and pending transactions to ensure synchronization across the network. This decentralized design eliminates the need for a central authority, reduces the risk of single points of failure, and enhances system reliability. Even if one node becomes compromised or goes offline, the remaining nodes continue to function normally.

These nodes work independently, share new blocks with each other and validate transactions before accepting them

4. Consensus Mechanism

To maintain consistency and agreement among all nodes, blockchain relies on a consensus mechanism. In this system, the **Longest Chain Rule** is used, which states that if multiple versions of the blockchain exist, the chain with the greatest number of blocks is considered the valid one. This approach assumes that the longest chain represents the most accumulated computational work and is therefore the most trustworthy.

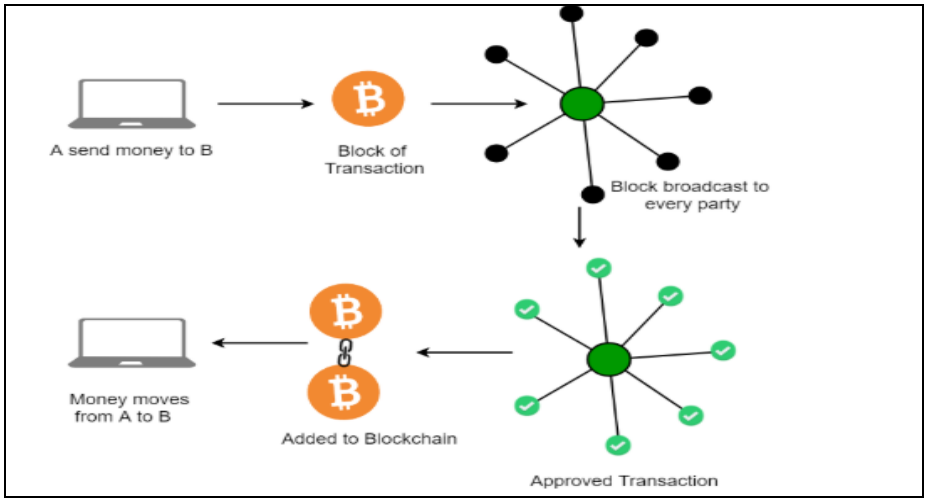
When nodes detect different blockchain versions, they compare them and adopt the longest valid chain. This mechanism prevents conflicts, resolves forks, and ensures that all participants eventually agree on a single, unified transaction history.

5. Transactions and Mining Reward

A blockchain transaction consists of three key components: sender, receiver, and amount. These transactions are initially stored in a pool of pending transactions until they are included in a newly mined block. When a miner successfully mines a block, all selected pending transactions are added to that block.

Additionally, a special reward transaction is automatically generated to compensate the miner for their effort. This reward serves as an incentive for miners to participate in the network and contribute computational power. Over time, as mining difficulty increases, transaction fees also become an important component of miner compensation.

In a decentralized blockchain network, transactions become valid and permanent only after being confirmed through mining and added to the blockchain. This process ensures security, transparency, and trust without the need for a central authority. Once a block is added to the chain, the recorded transactions cannot be altered or deleted, making the system tamper-resistant. This mechanism maintains the integrity of the network while enabling secure peer-to-peer digital transactions.



Example:
If Alice sends $100 to Bob, this transaction remains pending until a miner includes it in a block. When the block is mined, the miner might receive a reward like “Miner → Miner: 6.25 BTC” along with Alice’s transaction

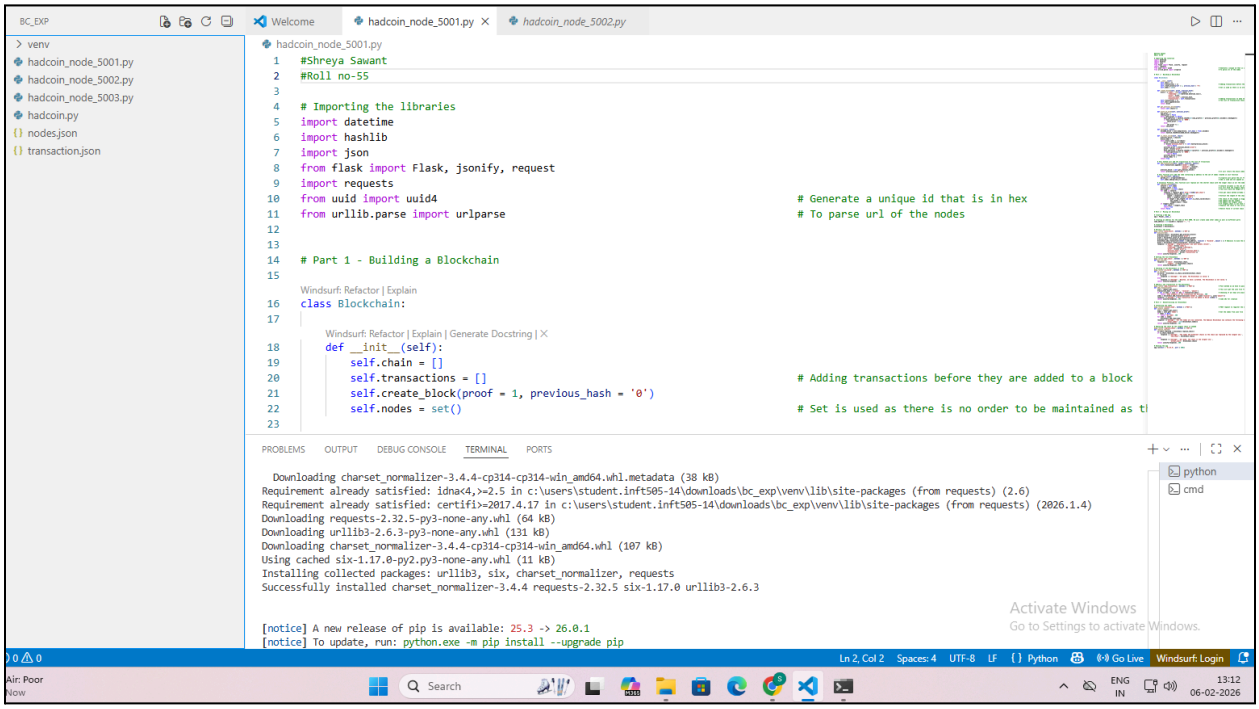
6. Chain Replacement

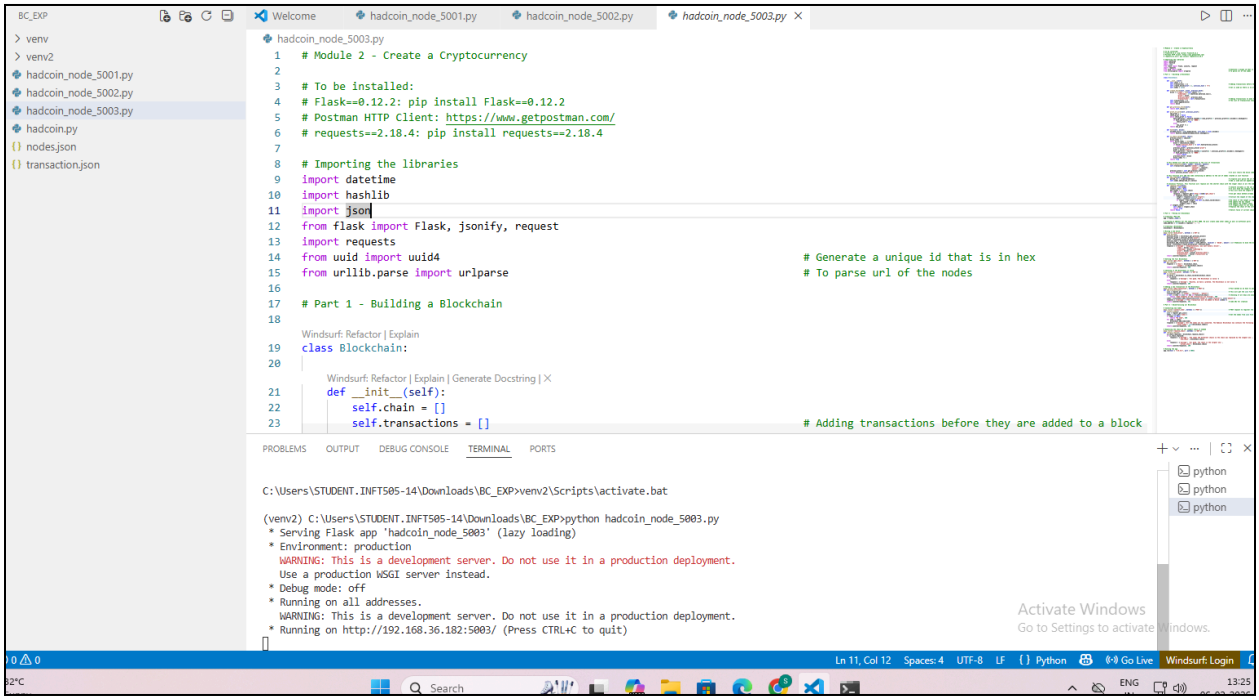
Chain replacement is a mechanism that ensures consistency and synchronization among blockchain nodes. When a node calls the /replace_chain endpoint, it requests blockchain data from its connected peers. If it discovers a longer and valid chain, it replaces its current chain with the updated version.

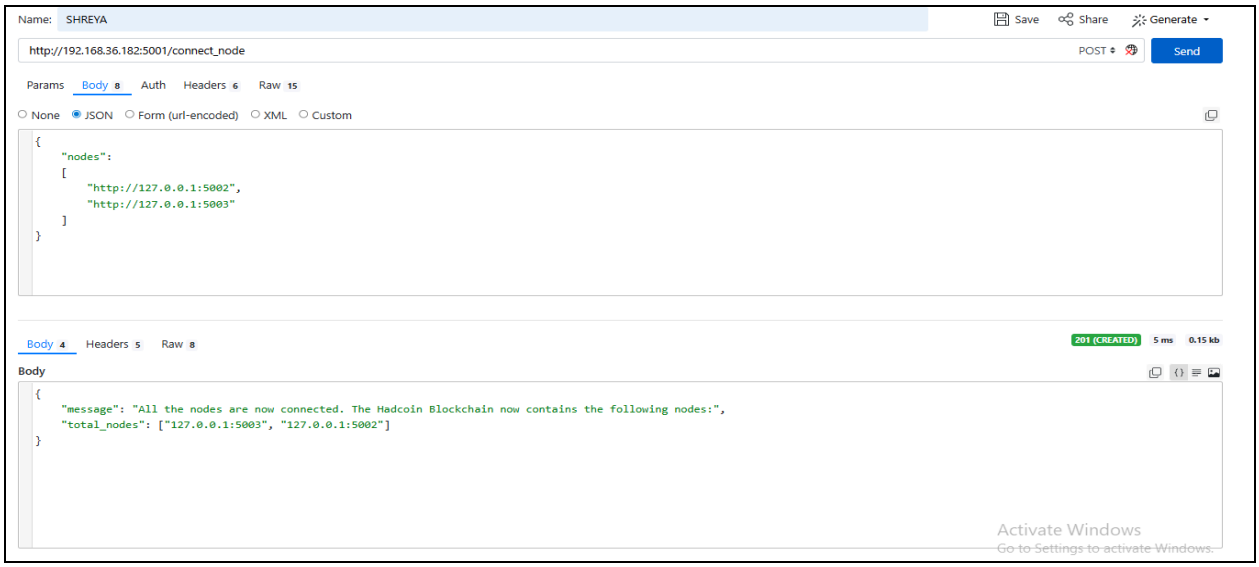
This process prevents discrepancies, resolves conflicts, and ensures that all nodes maintain the most accurate and up-to-date version of the blockchain. Chain replacement helps maintain a single authoritative ledger across the network, reinforcing the reliability and integrity of the decentralized system.

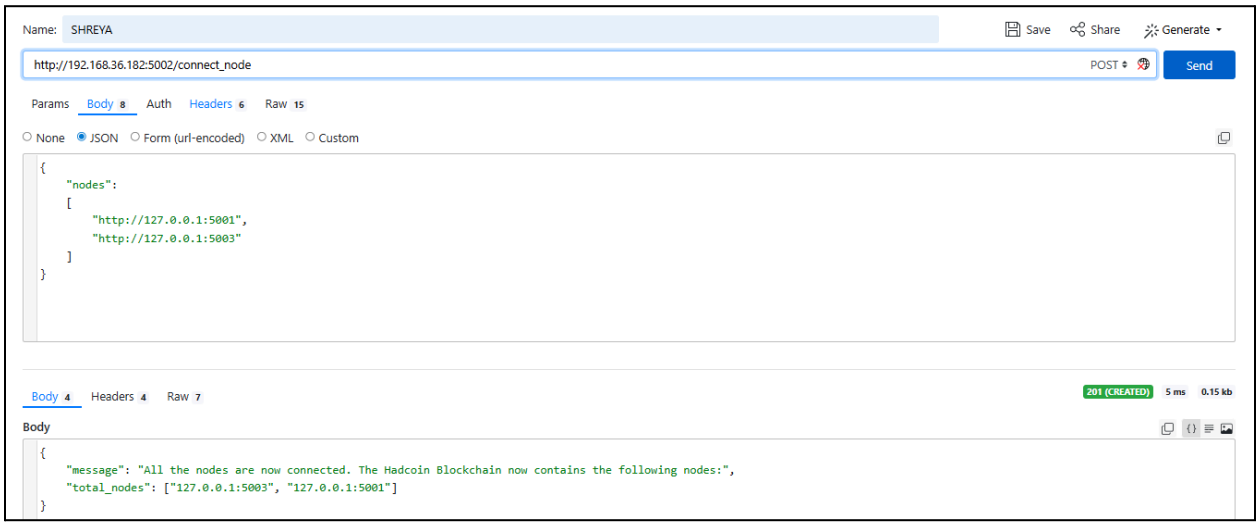
To handle this, the chain replacement mechanism follows the “Longest Chain Rule,” assuming that the chain with the most blocks contains the highest amount of computational work and is therefore the most reliable. Before replacing its chain, a node also verifies the validity of the longer chain to prevent acceptance of incorrect or manipulated data. This process ensures decentralization, consistency, and security across the entire blockchain network.

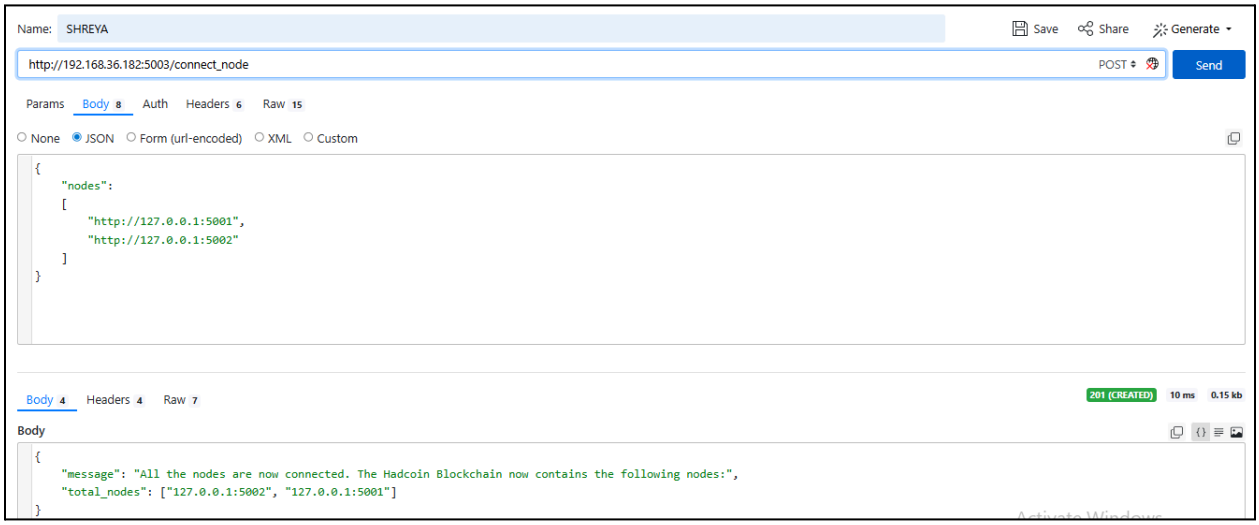
Output -

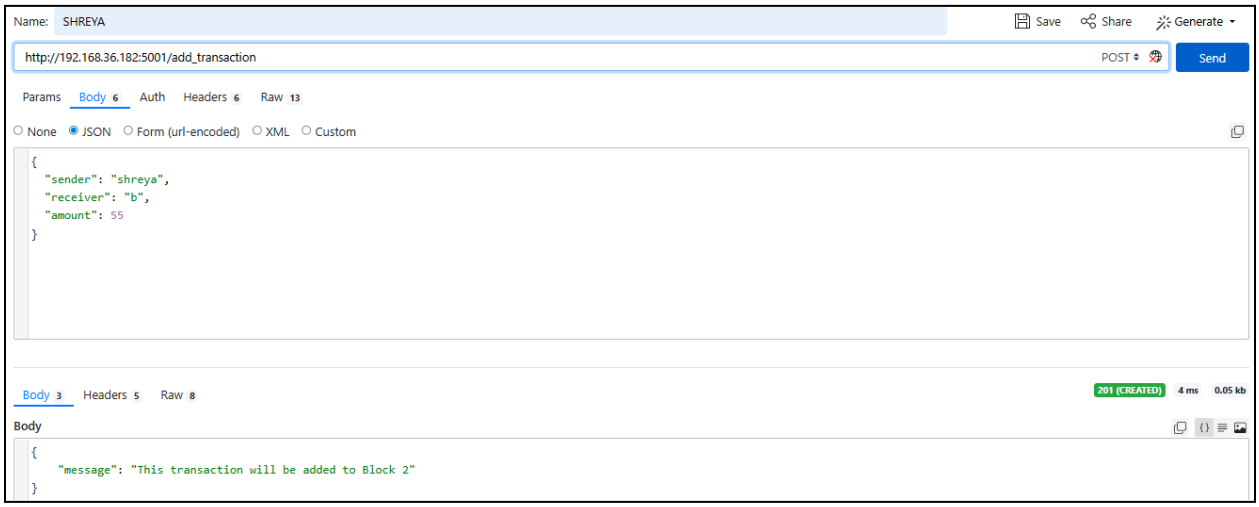

BC_EXP
venv
hadcoin_node_5001.py
hadcoin_node_5002.py
hadcoin_node_5003.py
hadcoin.py
nodes.json
transaction.json
Welcome
hadcoin_node_5001.py
hadcoin_node_5002.py
#Shreya Sawant
#Roll no-55
# Importing the libraries
import datetime
import hashlib
import json
from flask import Flask, jsonify, request
import requests
from uuid import uuid4 # Generate a unique id that is in hex
from urllib.parse import urlparse # To parse url of the nodes
# Part 1 - Building a Blockchain
Windsurf: Refactor | Explain
class Blockchain:
Windsurf: Refactor | Explain | Generate Docstring
def __init__(self):
self.chain = []
self.transactions = [] # Adding transactions before they are added to a block
self.create_block(proof = 1, previous_hash = '0')
self.nodes = set() # Set is used as there is no order to be maintained as t
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Downloading charset_normalizer-3.4.4-cp314-cp314-win_amd64.whl.metadata (38 kB)
Requirement already satisfied: idna<4,>=2.5 in c:\users\student.inft505-14\downloads\bc_exp\venv\lib\site-packages (from requests) (2.6)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\student.inft505-14\downloads\bc_exp\venv\lib\site-packages (from requests) (2026.1.4)
Downloading requests-2.32.5-py3-none-any.whl (64 kB)
Downloading urllib3-2.6.3-py3-none-any.whl (131 kB)
Downloading charset_normalizer-3.4.4-cp314-cp314-win_amd64.whl (107 kB)
Using cached six-1.17.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: urllib3, six, charset_normalizer, requests
Successfully installed charset_normalizer-3.4.4 requests-2.32.5 six-1.17.0 urllib3-2.6.3
[notice] A new release of pip is available: 25.3 -> 26.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
python
cmd
Activate Windows
Go to Settings to activate Windows.
Ln 2, Col 2 Spaces: 4 UTF-8 LF Python Go Live Windsurf: Login
Search
ENG IN
13:12
06-02-2026

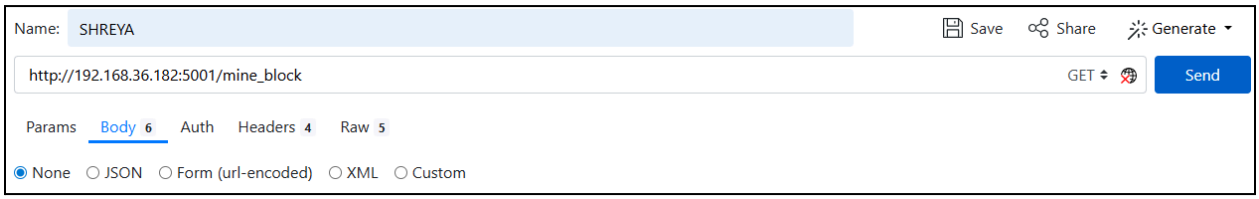

BC_EXP
venv
venv2
hadcoin_node_5001.py
hadcoin_node_5002.py
hadcoin_node_5003.py
hadcoin.py
nodes.json
transaction.json
Welcome
hadcoin_node_5001.py
hadcoin_node_5002.py
hadcoin_node_5003.py
# Module 2 - Create a Cryptocurrency
# To be installed:
# Flask==0.12.2: pip install Flask==0.12.2
# Postman HTTP Client: https://www.getpostman.com/
# requests==2.18.4: pip install requests==2.18.4
# Importing the libraries
import datetime
import hashlib
import json
from flask import Flask, jsonify, request
import requests
from uuid import uuid4 # Generate a unique id that is in hex
from urllib.parse import urlparse # To parse url of the nodes
# Part 1 - Building a Blockchain
Windsurf: Refactor | Explain
class Blockchain:
Windsurf: Refactor | Explain | Generate Docstring
def __init__(self):
self.chain = []
self.transactions = [] # Adding transactions before they are added to a block
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
C:\Users\STUDENT.INFT505-14\Downloads\BC_EXP>venv2\Scripts\activate.bat
(venv2) C:\Users\STUDENT.INFT505-14\Downloads\BC_EXP>python hadcoin_node_5003.py
* Serving Flask app 'hadcoin_node_5003' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://192.168.36.182:5003/ (Press CTRL+C to quit)
python
python
python
Activate Windows
Go to Settings to activate Windows.
Ln 11, Col 12 Spaces: 4 UTF-8 LF Python Go Live Windsurf: Login
Search
ENG
13:25

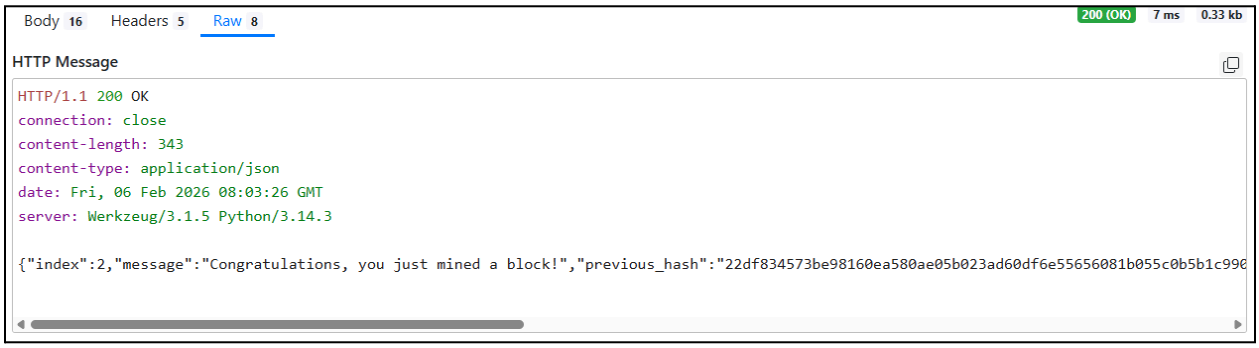

Name: SHREYA
Save
Share
Generate
http://192.168.36.182:5001/connect_node
POST
Send
Params
Body 8
Auth
Headers 6
Raw 15
None
JSON
Form (url-encoded)
XML
Custom
{
"nodes":
[
"http://127.0.0.1:5002",
"http://127.0.0.1:5003"
]
}
Body 4
Headers 5
Raw 8
201 (CREATED)
5 ms
0.15 kb
Body
{
"message": "All the nodes are now connected. The Hadcoin Blockchain now contains the following nodes:",
"total_nodes": ["127.0.0.1:5003", "127.0.0.1:5002"]
}
Activate Windows
Go to Settings to activate Windows.


Name: SHREYA
Save
Share
Generate
http://192.168.36.182:5002/connect_node
POST
Send
Params
Body 8
Auth
Headers 6
Raw 15
None
JSON
Form (url-encoded)
XML
Custom
{
"nodes":
[
"http://127.0.0.1:5001",
"http://127.0.0.1:5003"
]
}
Body 4
Headers 4
Raw 7
201 (CREATED)
5 ms
0.15 kb
Body
{
"message": "All the nodes are now connected. The Hadcoin Blockchain now contains the following nodes:",
"total_nodes": ["127.0.0.1:5003", "127.0.0.1:5001"]
}


Name: SHREYA
Save
Share
Generate
http://192.168.36.182:5003/connect_node
POST
Send
Params
Body 8
Auth
Headers 6
Raw 15
None
JSON
Form (url-encoded)
XML
Custom
{
"nodes":
[
"http://127.0.0.1:5001",
"http://127.0.0.1:5002"
]
}
Body 4
Headers 4
Raw 7
201 (CREATED)
10 ms
0.15 kb
Body
{
"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"]
}



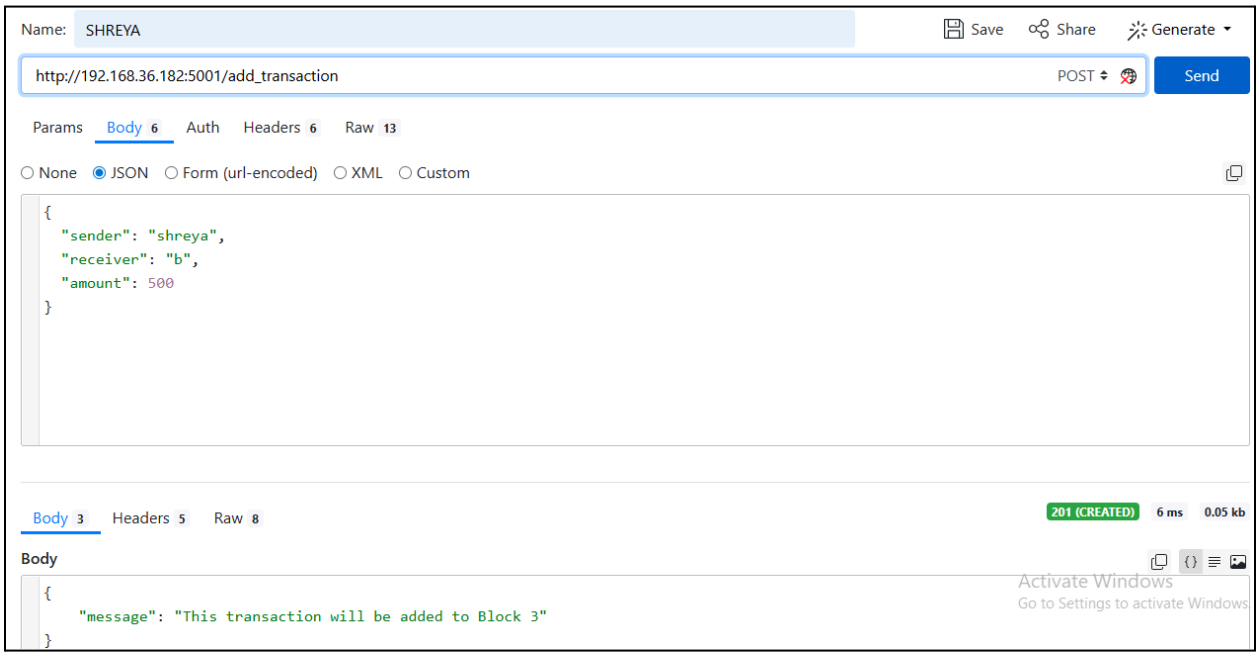


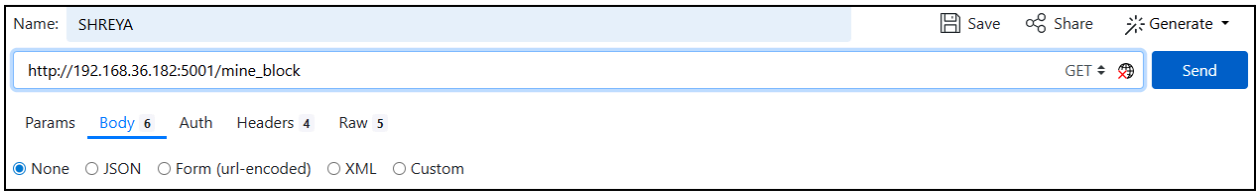


Body -

{
"index": 2,
"message": "Congratulations, you just mined a block!",
"previous_hash":
"22df834573be98160ea580ae05b023ad60df6e55656081b055c0b5b1c99069c1",
"proof": 533,
"timestamp": "2026-02-06 13:33:26.709359",
"transactions": [{
"amount": 55,
"receiver": "b",
"sender": "shreya"
}, {

```
        "amount": 1,
        "receiver": "Richard",
        "sender": "e41582e3bc3946088356875f4814fdf7"
    }]
}
```

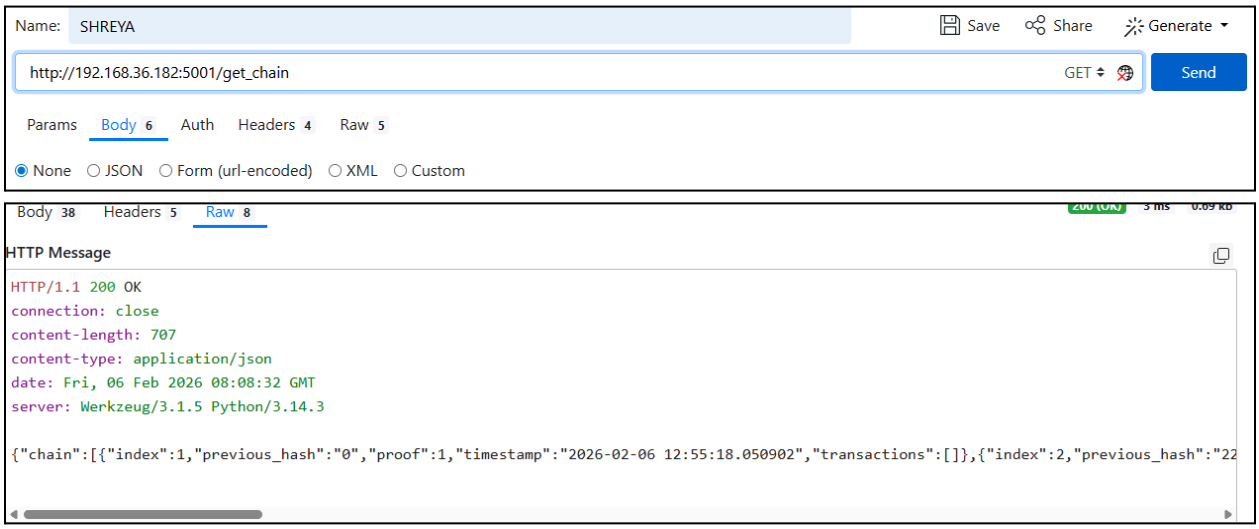





Body -

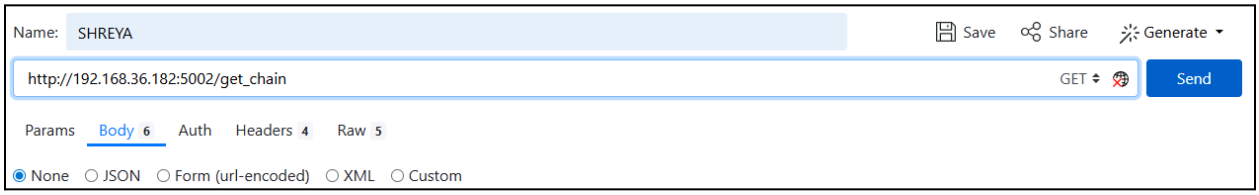
```
{
    "index": 3,
    "message": "Congratulations, you just mined a block!",
    "previous_hash":
"d1c264bae4f31a4e4ac7dfd22c4119bb2e4e18efcb8d40e94e471503a56501e1",
    "proof": 45293,
    "timestamp": "2026-02-06 13:37:37.379214",
    "transactions": [{
        "amount": 500,
        "receiver": "b",
        "sender": "shreya"
    }, {
```

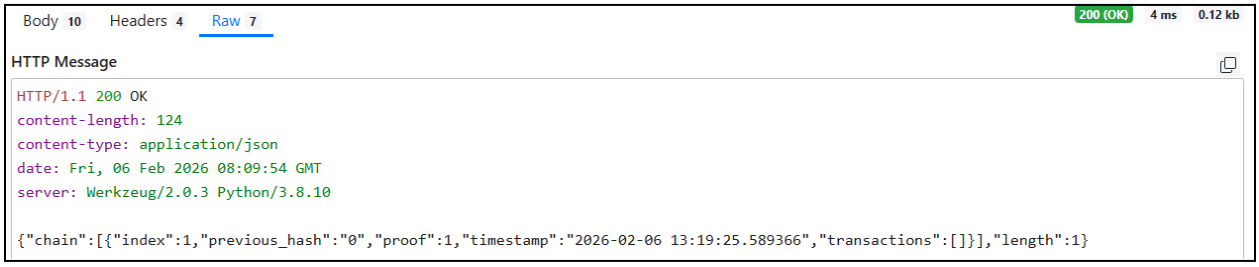
```
        "amount": 1,
        "receiver": "Richard",
        "sender": "e41582e3bc3946088356875f4814fdf7"
    }]
}
```


```
Body -{
    "chain": [{
        "index": 1,
        "previous_hash": "0",
        "proof": 1,
        "timestamp": "2026-02-06 12:55:18.050902",
        "transactions": []
    }, {
        "index": 2,
        "previous_hash":
"22df834573be98160ea580ae05b023ad60df6e55656081b055c0b5b1c99069c1",
        "proof": 533,
        "timestamp": "2026-02-06 13:33:26.709359",
        "transactions": [{
            "amount": 55,
            "receiver": "b",
            "sender": "shreya"
        }, {
            "amount": 1,
            "receiver": "Richard",
            "sender": "e41582e3bc3946088356875f4814fdf7"
        }]
    }, {
        "index": 3,
```

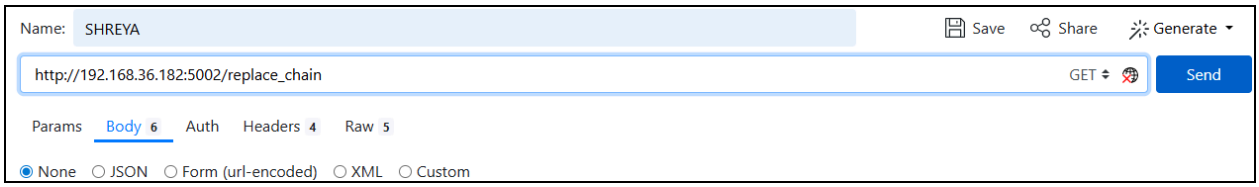
```
        "previous_hash":
"d1c264bae4f31a4e4ac7dfd22c4119bb2e4e18efcb8d40e94e471503a56501e1",
        "proof": 45293,
        "timestamp": "2026-02-06 13:37:37.379214",
        "transactions": [{
            "amount": 500,
            "receiver": "b",
            "sender": "shreya"
        }, {
            "amount": 1,
            "receiver": "Richard",
            "sender": "e41582e3bc3946088356875f4814fdf7"
        }]
    }],
    "length": 3
}
```

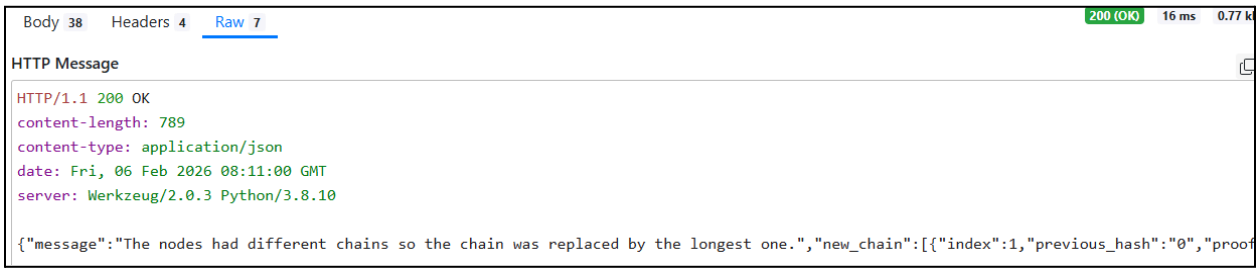





Body-


```
{
    "chain": [{
        "index": 1,
        "previous_hash": "0",
        "proof": 1,
        "timestamp": "2026-02-06 13:19:25.589366",
        "transactions": []
    }],
    "length": 1
}
```

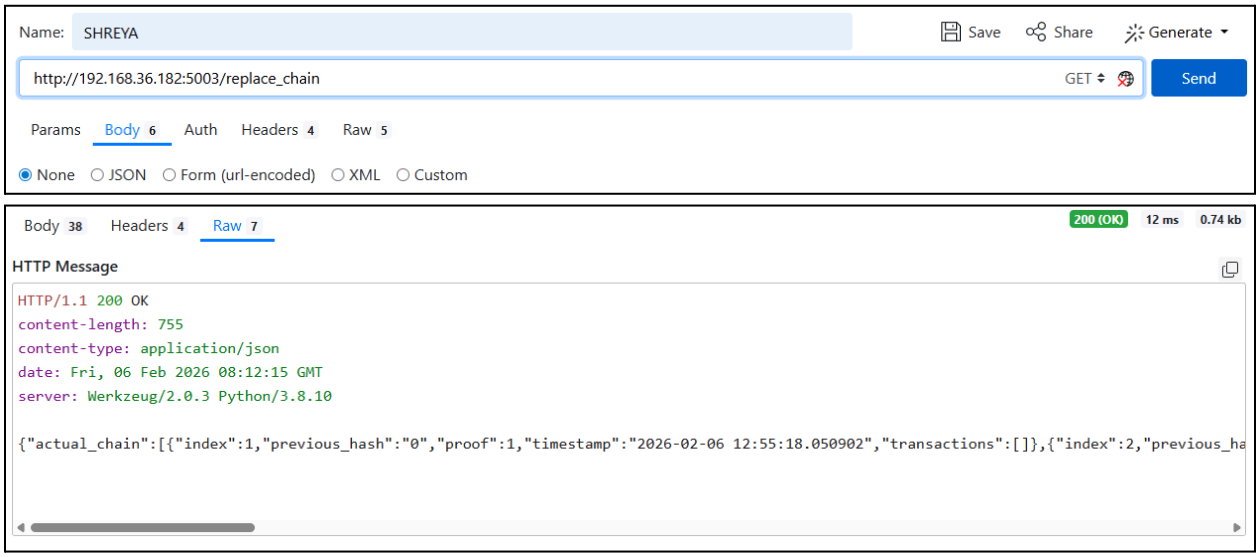




body-\


```
{
    "message": "The nodes had different chains so the chain was replaced by the longest one.",
    "new_chain": [{
        "index": 1,
        "previous_hash": "0",
        "proof": 1,
        "timestamp": "2026-02-06 12:55:18.050902",
        "transactions": []
    }, {
        "index": 2,
        "previous_hash": "22df834573be98160ea580ae05b023ad60df6e55656081b055c0b5b1c99069c1",
        "proof": 533,
        "timestamp": "2026-02-06 13:33:26.709359",
        "transactions": [{
            "amount": 55,
            "receiver": "b",
            "sender": "shreya"
        }, {
            "amount": 1,
            "receiver": "Richard",
            "sender": "e41582e3bc3946088356875f4814fdf7"
        }]
    }, {
        "index": 3,
        "previous_hash": "d1c264bae4f31a4e4ac7dfd22c4119bb2e4e18efcb8d40e94e471503a56501e1",
        "proof": 45293,
        "timestamp": "2026-02-06 13:37:37.379214",
        "transactions": [{
            "amount": 500,
```

```
            "receiver": "b",
            "sender": "shreya"
        }, {
            "amount": 1,
            "receiver": "Richard",
            "sender": "e41582e3bc3946088356875f4814fdf7"
        }]
    }]
}
```


```
body-{
    "actual_chain": [{
        "index": 1,
        "previous_hash": "0",
        "proof": 1,
        "timestamp": "2026-02-06 12:55:18.050902",
        "transactions": []
    }, {
        "index": 2,
        "previous_hash":
"22df834573be98160ea580ae05b023ad60df6e55656081b055c0b5b1c99069c1",
        "proof": 533,
        "timestamp": "2026-02-06 13:33:26.709359",
        "transactions": [{
            "amount": 55,
            "receiver": "b",
            "sender": "shreya"
        }, {
            "amount": 1,
```

```
            "receiver": "Richard",
            "sender": "e41582e3bc3946088356875f4814fdf7"
        }]
    }, {
        "index": 3,
        "previous_hash":
"d1c264bae4f31a4e4ac7dfd22c4119bb2e4e18efcb8d40e94e471503a56501e1",
        "proof": 45293,
        "timestamp": "2026-02-06 13:37:37.379214",
        "transactions": [{
            "amount": 500,
            "receiver": "b",
            "sender": "shreya"
        }, {
            "amount": 1,
            "receiver": "Richard",
            "sender": "e41582e3bc3946088356875f4814fdf7"
        }]
    }],
    "message": "All good. The chain is the largest one."
}
```


Conclusion -

In this experiment, we mainly worked with APIs and learned how to interact with the blockchain using Postman. We used different API endpoints to connect nodes, add transactions, mine blocks, fetch the blockchain, and replace it with the longest chain. This helped us understand how blockchain functions can be accessed and controlled through API calls in a real-world application.