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

Guidelines

Lab Objectives: To implement public and private Blockchain.

Lab Outcomes (LO): Demonstrate the concept of Blockchain in real-world Applications (LO4)

Task to be performed :

1. Download the code from folder, Lab_3
2. Install requests in the virtual environment created in the Lab 2. (Follow the instructions)
3. Run the files - hadcoin_node_5001.py, hadcoin_node_5002.py, hadcoin_node_5003.py in 3 different terminals.
4. Open Postman, from each node - invoke connect_node() and pass the peers as POST requests.
5. Perform the following functions
    - Add Transactions - invoke add_transactions() as a POST request.
    - mining - mine_block(),
    - fetch the chain - get_chain(),
    - replace the longest chain - replace_chain()
6. Modify the code such that transactions are removed after they are added to the block. Tools & Libraries used :
    - Install Flask : pip install Flask
    - Download Postman from https://www.postman.com/
    - Python Libraries : datetime, jsonify, hashlib, uuid4, urlparse, request
    - Install requests : pip install requests==2.18.4 Instructions : (Prepare for viva for the following topics)

1. Challenges in P2P networks
2. How transactions are performed on the network?
3. Explain the role of mempools
4. Write briefly about the libraries and the tools used during implementation.

Outcome :

1. Understood the challenges in P2P networks, how transactions are performed and how a miner mines a block to be added in a blockchain.
2. Implemented a Cryptocurrency in Python using Flask, Postman and Python libraries such as datetime, jsonify, hashlib, uuid4, urlparse, request.
3. Successfully mined the blocks among a P2P network with 3 nodes.
4. Performed transactions via the network.
5. Successfully updated the block across the network
6. Prepare a document with Aim, Tasks performed, Program, Output and Conclusion.
7. Submit the hardcopy by the 2nd week of August 2023 (As per the instructions, submit a hard copy of the same).

Theory:

1. Blockchain Overview

Blockchain is a **distributed and decentralized ledger** that stores information in a series of linked blocks. Each block contains:

- Transaction data
- Timestamp
- Previous block's hash
- Its own unique hash (digital fingerprint)

Once data is recorded in a blockchain, it becomes **immutable** because altering one block would require recalculating all subsequent blocks.

2. Mining

Mining is the process of:

1. Collecting pending transactions into a block.
2. Performing a computational puzzle (Proof-of-Work) to find a valid hash.
3. Adding the new block to the blockchain. Broadcasting it to all connected peers.

Miners are rewarded with cryptocurrency for successfully mining a block.

3. Multi-Node Blockchain Network

In this lab, we simulate **three independent blockchain nodes** (`5001`, `5002`, `5003`).

Each node:

- Runs on a separate port.
- Maintains its own copy of the blockchain.
- Can connect with peers to share and validate blocks.

4. Consensus Mechanism

We use the **Longest Chain Rule**:

- If multiple versions of the chain exist, the **longest valid chain** is chosen.
- This ensures all nodes agree on a single transaction history.

5. Transactions & Mining Reward

Each transaction has:

- Sender
- Receiver
- Amount

When mining a block:

- Pending transactions are added to the block.
- A **reward transaction** is added automatically to pay the miner.

6. Chain Replacement

When `/replace_chain` is called:

1. Node requests chains from peers.
2. If it finds a longer and valid chain, it replaces its own.
3. This keeps the blockchain consistent across all nodes.

Tools & Libraries Used

- **Python 3.x**
- **Flask** – Web framework for API endpoints

  ```
  pip install Flask
  ```

- **Requests** – For HTTP communication between nodes

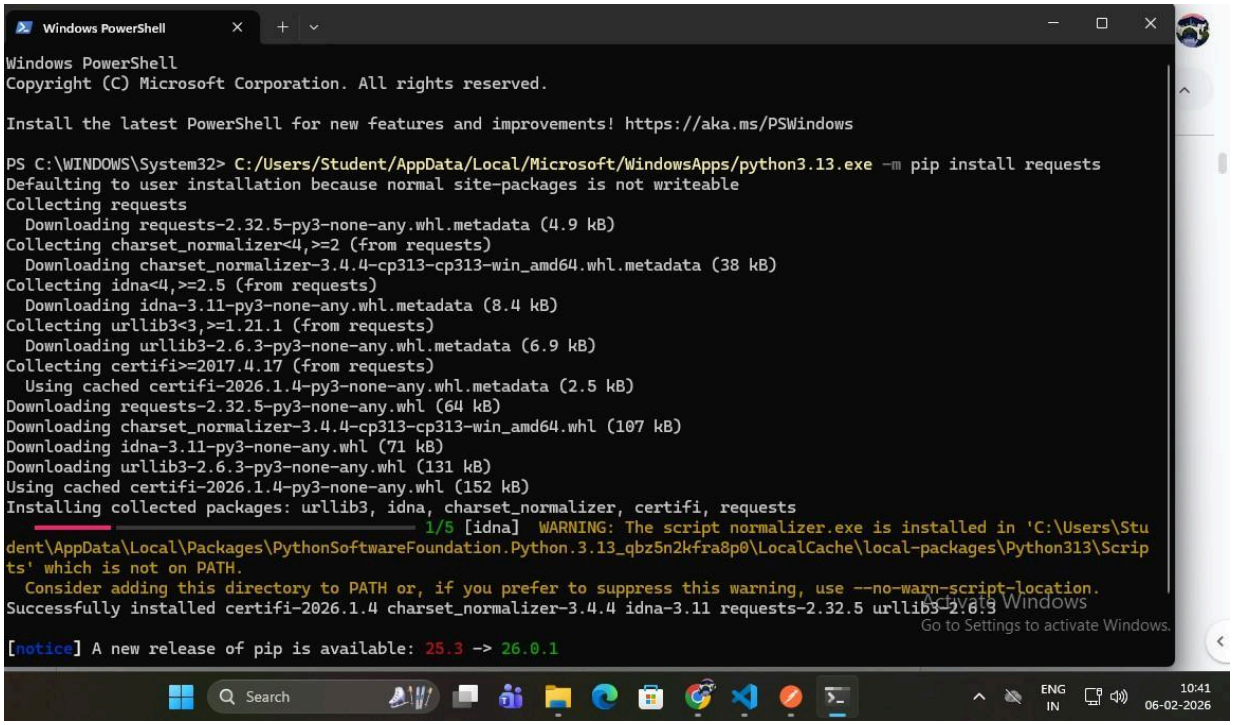
  ```
  pip install requests==2.18.4
  ```

- **Postman** – For testing API requests
- Python Standard Libraries:
  - `datetime`
  - `jsonify`

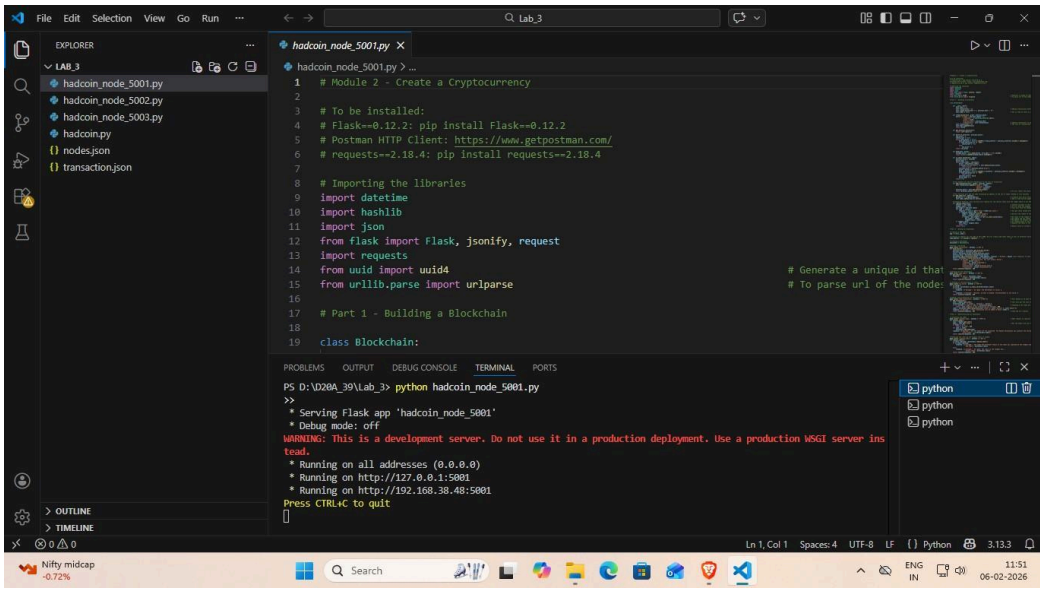
- `hashlib`
- `uuid4`
- `urlparse`
- `request`

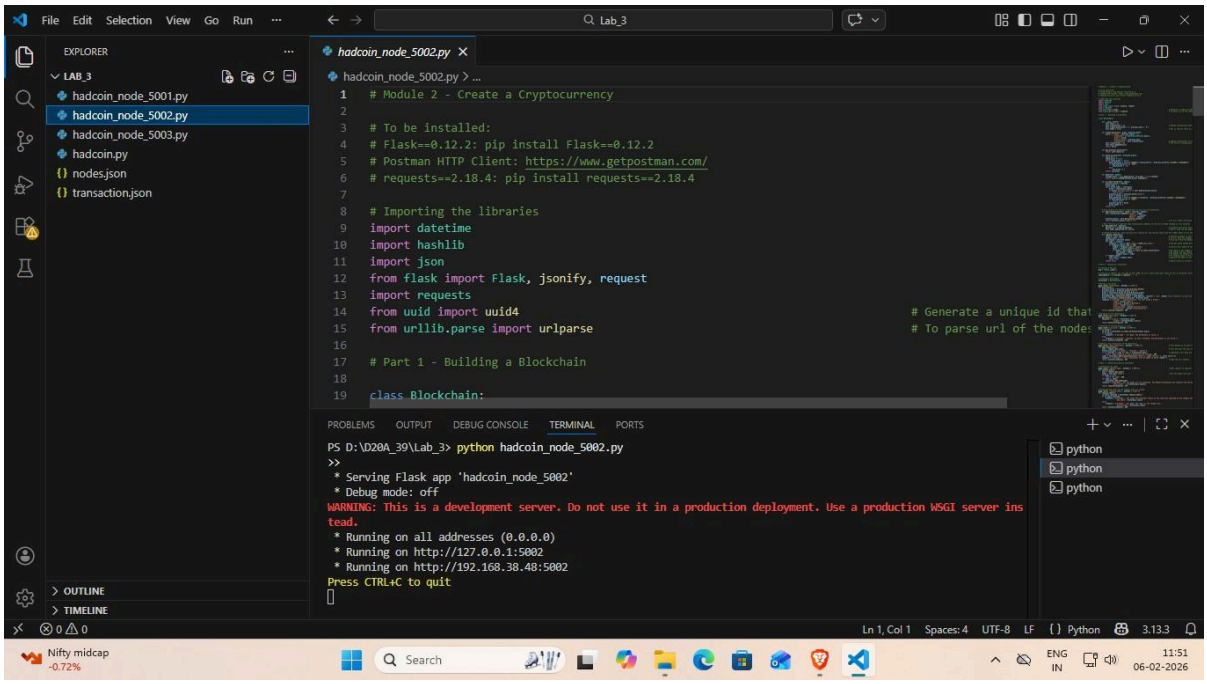
Procedure and Output:

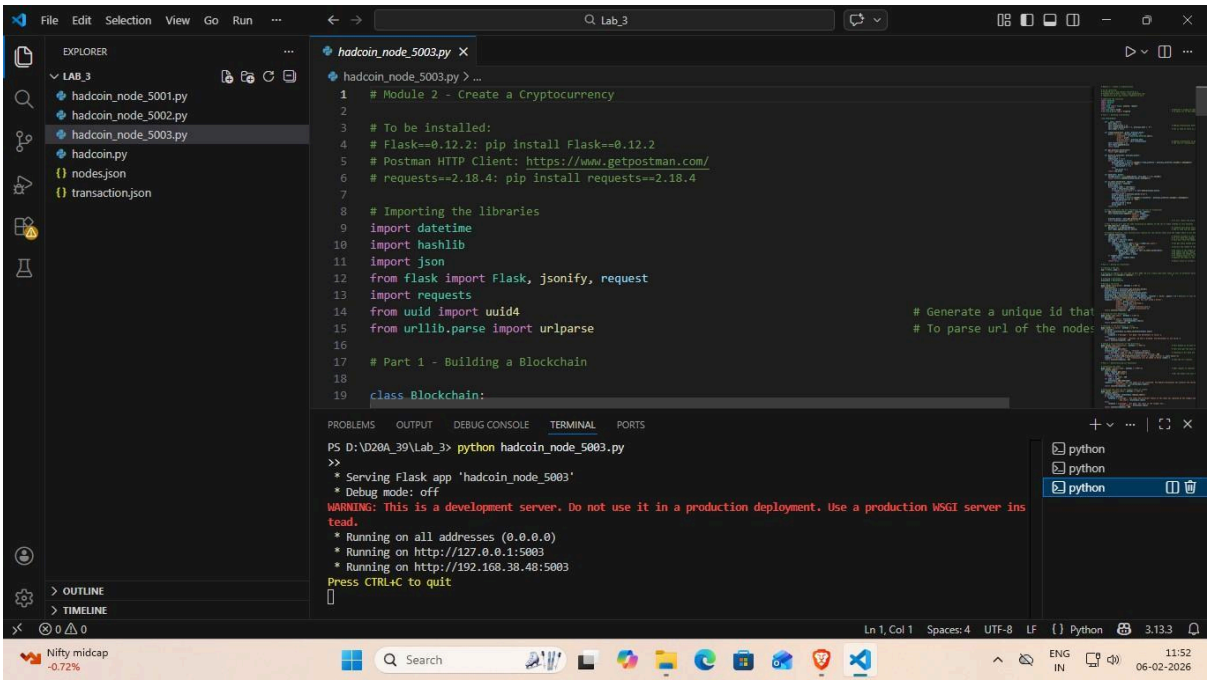
1. Download `hadcoin_node_5001.py`, `hadcoin_node_5002.py`, `hadcoin_node_5003.py`.
2. Install required packages.



3. Run each node in separate terminals.



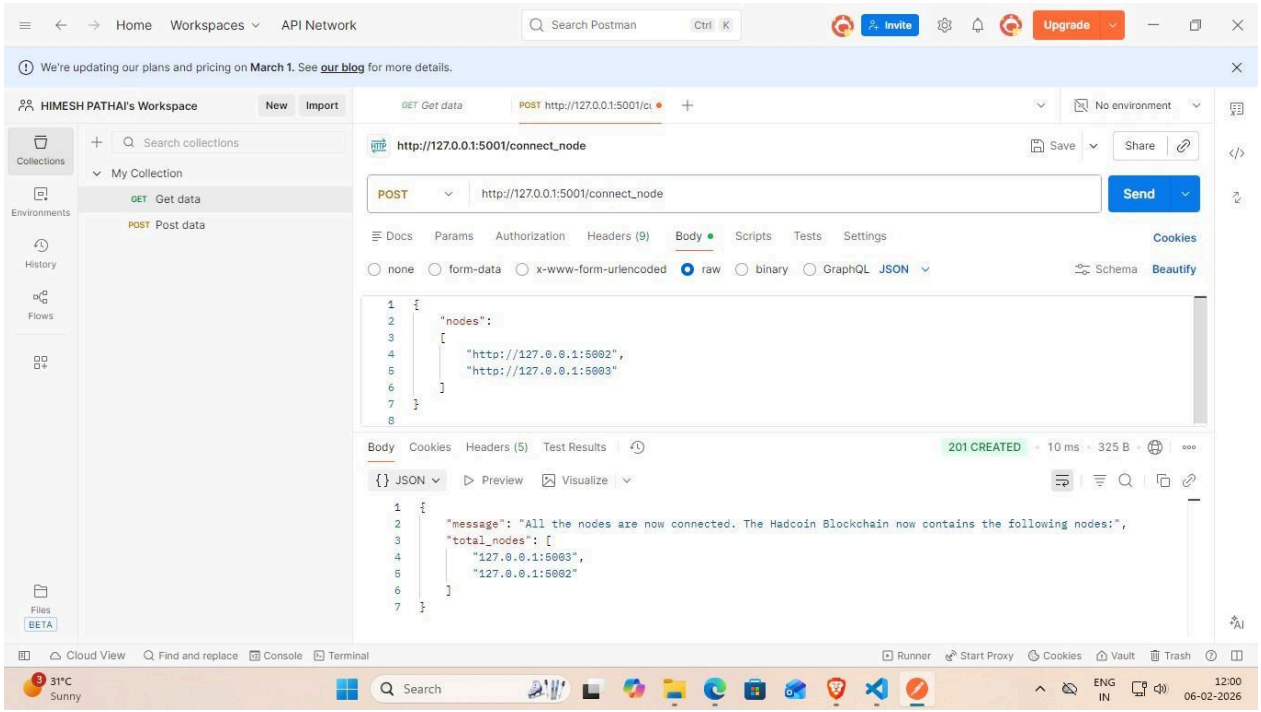


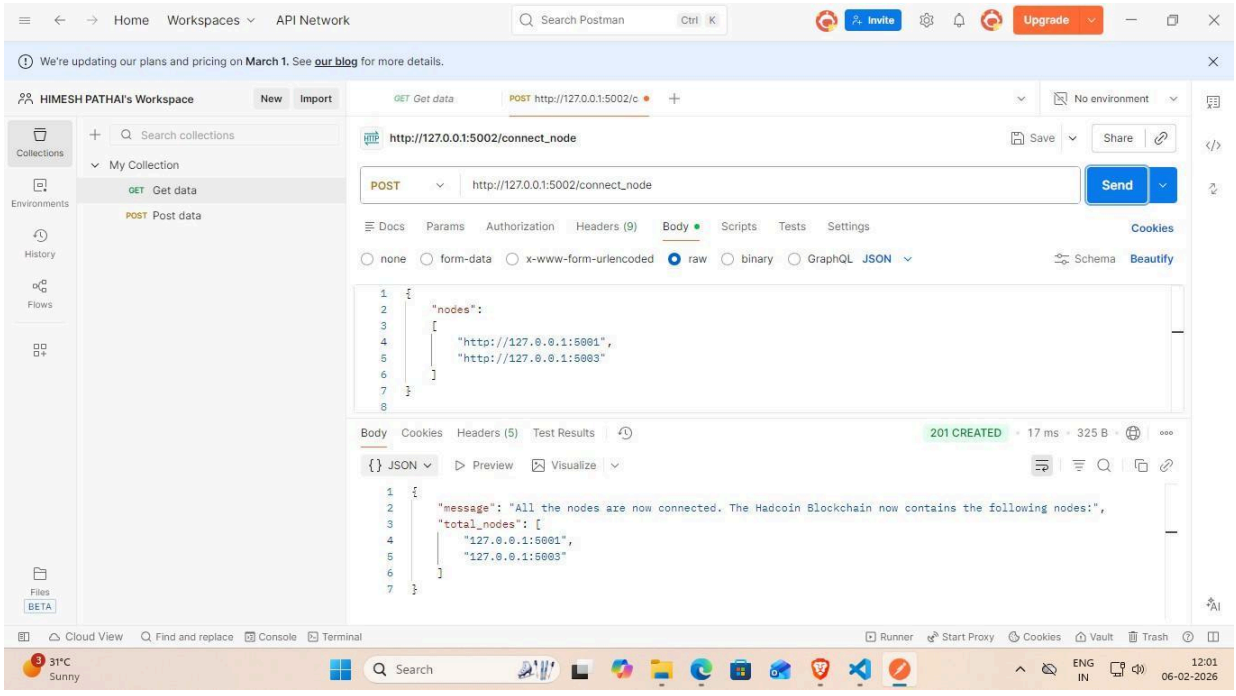


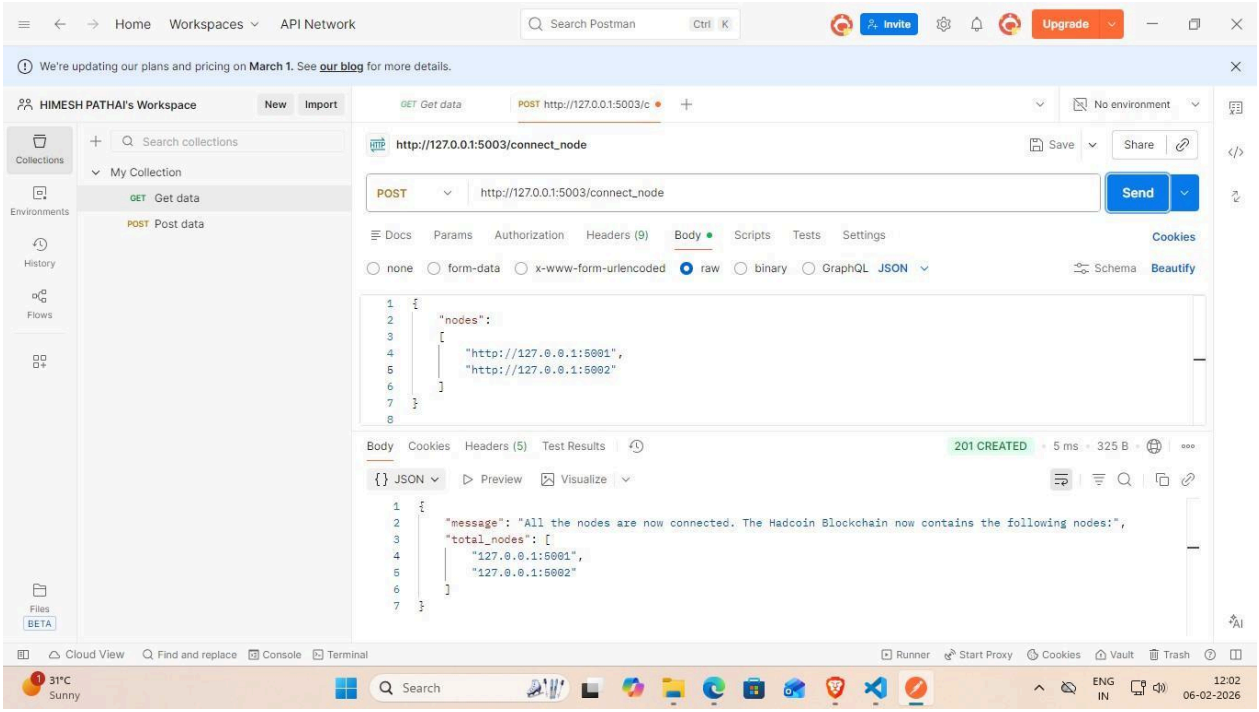
4. Connect nodes using Postman – POST `/connect_node`.


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

5.


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
We're updating our plans and pricing on March 1. See our blog for more details.
HIMESH PATHAI's Workspace
New
Import
Collections
Environments
History
Flows
Files
BETA
Search collections
My Collection
GET Get data
POST Post data
GET Get data
POST http://127.0.0.1:5001/c
No environment
http://127.0.0.1:5001/connect_node
Save
Share
POST
http://127.0.0.1:5001/connect_node
Send
Docs
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{
"nodes":
[
"http://127.0.0.1:5002",
"http://127.0.0.1:5003"
]
}
Body
Cookies
Headers (5)
Test Results
201 CREATED
10 ms
325 B
{} JSON
Preview
Visualize
{
"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"
]
}
Cloud View
Find and replace
Console
Terminal
Runner
Start Proxy
Cookies
Vault
Trash
31°C
Sunny
Search
ENG
IN
12:00
06-02-2026


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
We're updating our plans and pricing on March 1. See our blog for more details.
HIMESH PATHAI's Workspace
New
Import
Collections
Environments
History
Flows
Files
BETA
Search collections
My Collection
GET Get data
POST Post data
GET Get data
POST http://127.0.0.1:5002/c
No environment
http://127.0.0.1:5002/connect_node
Save
Share
POST
http://127.0.0.1:5002/connect_node
Send
Docs
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{
"nodes":
[
"http://127.0.0.1:5001",
"http://127.0.0.1:5003"
]
}
Body
Cookies
Headers (5)
Test Results
201 CREATED
17 ms
325 B
{} JSON
Preview
Visualize
{
"message": "All the nodes are now connected. The Hadcoin Blockchain now contains the following nodes:",
"total_nodes": [
"127.0.0.1:5001",
"127.0.0.1:5003"
]
}
Cloud View
Find and replace
Console
Terminal
Runner
Start Proxy
Cookies
Vault
Trash
31°C
Sunny
Search
ENG
IN
12:01
06-02-2026



6. Add transactions – POST
7. Add 3 transactions in 5001
8. Condition 2 — Each node must have **different chain lengths**

Example:

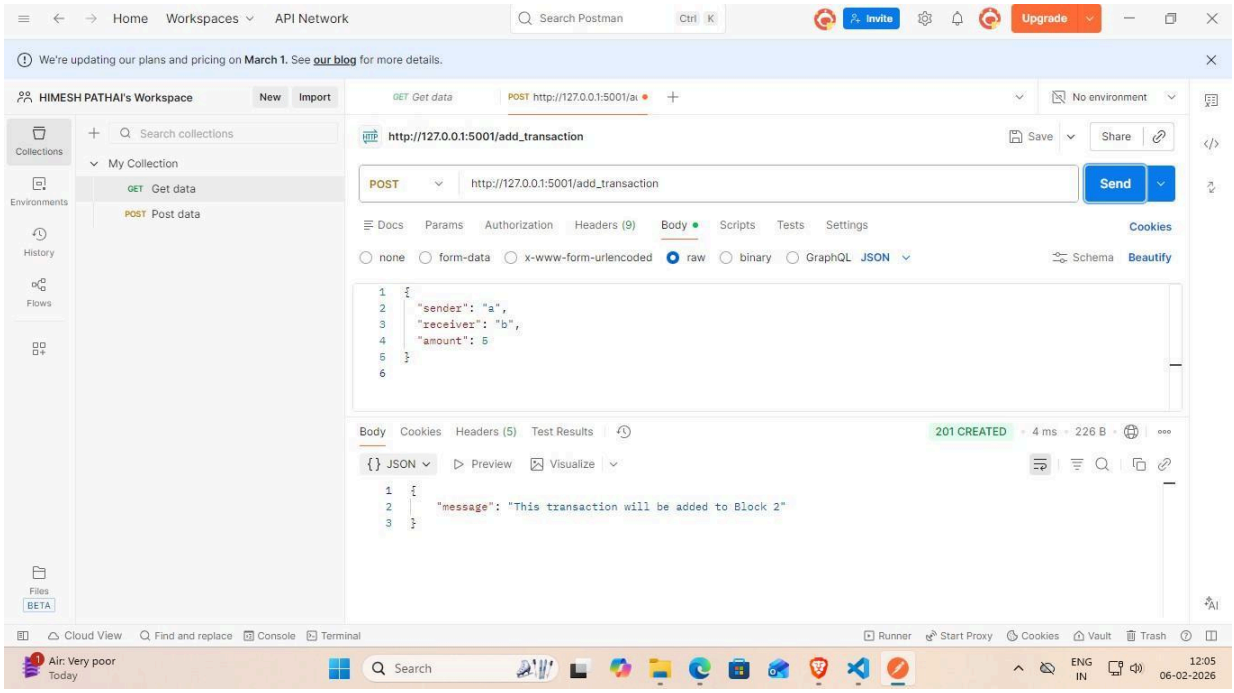
If you mined 1 block on 5001, but did not mine on 5002 or 5003 →

- **5001 chain length = 2**
- **5002 chain length = 1**
- **5003 chain length = 1**

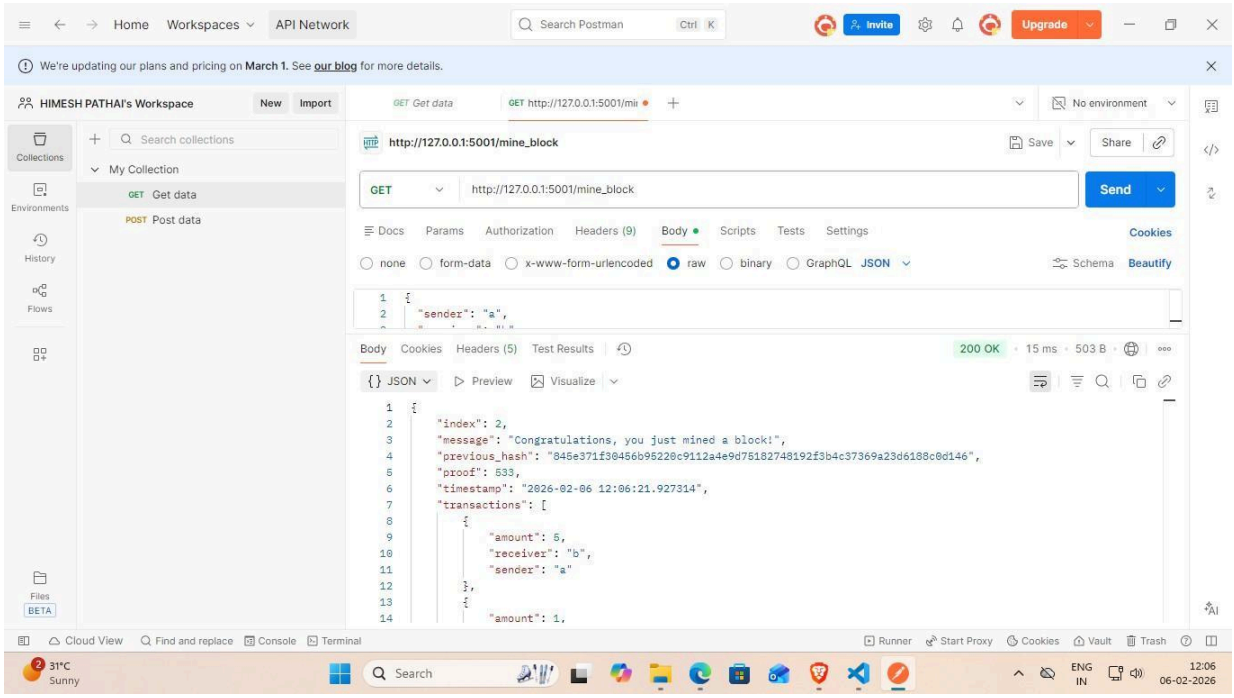
STEP 2 — Perform all transactions ONLY from one node (e.g., 5001)

POST → `http://127.0.0.1:5001/add_transaction`

```
{
  "sender": "a",
  "receiver": "b",
  "amount": 5
}
```

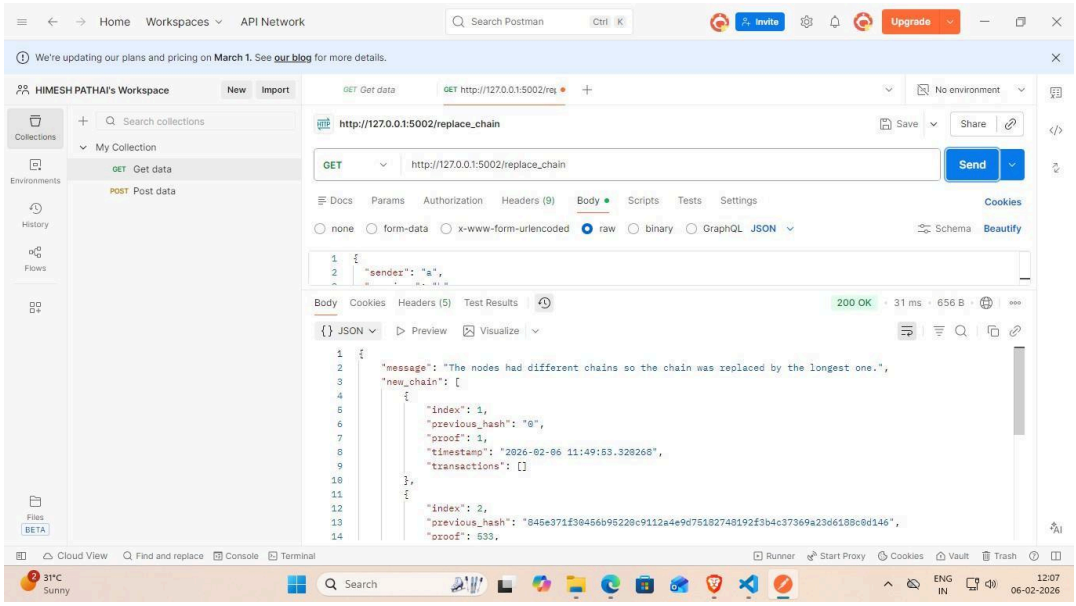


Then 5001 → GET `/mine_block`



STEP 3 — Now go to 5002 and run:

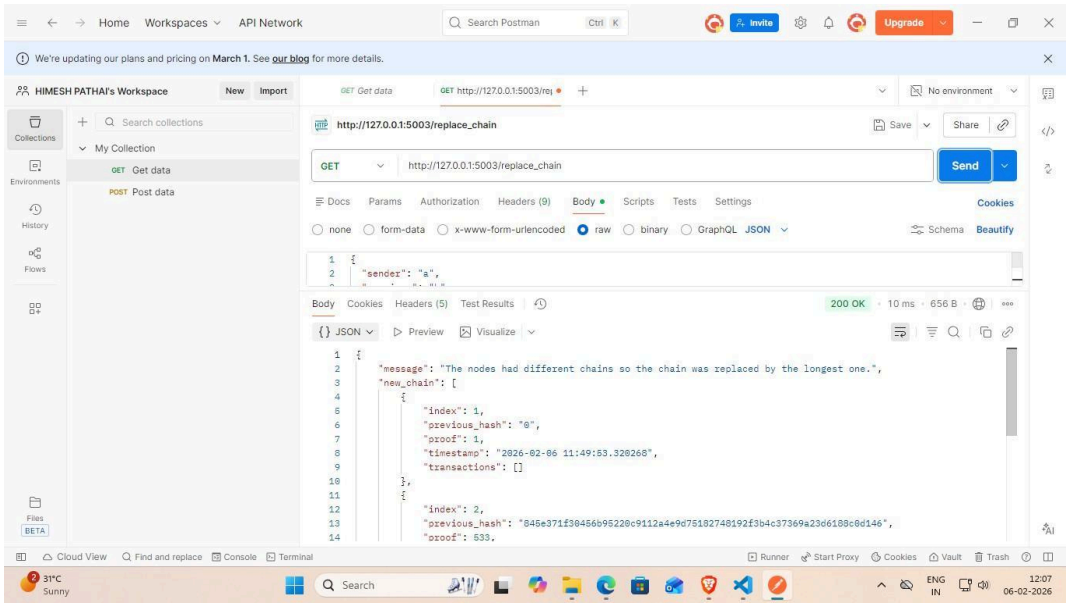
GET → `/replace_chain`



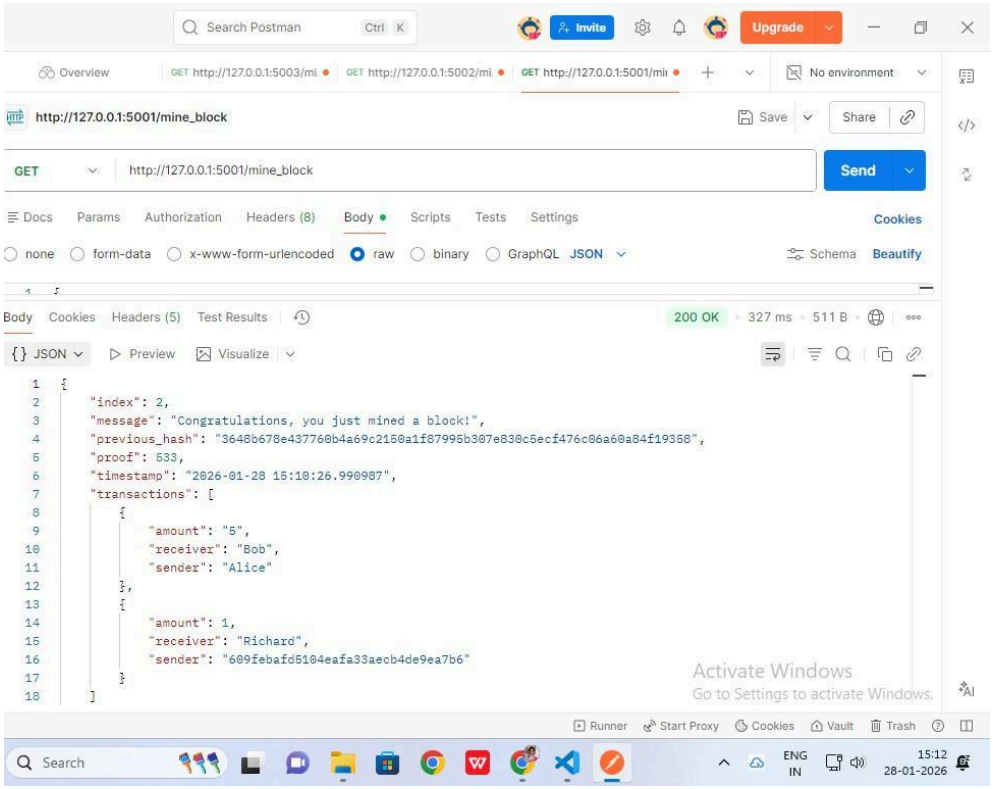
STEP 4 — Repeat on 5003

GET → `/replace_chain`

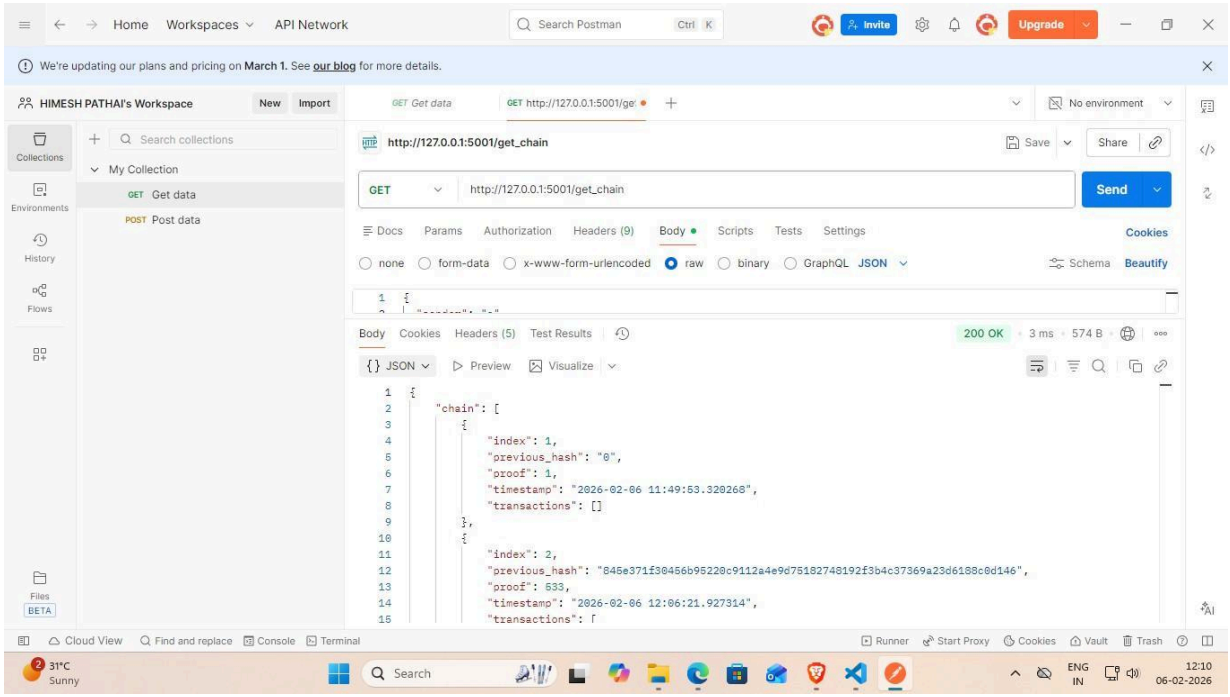
Same result.

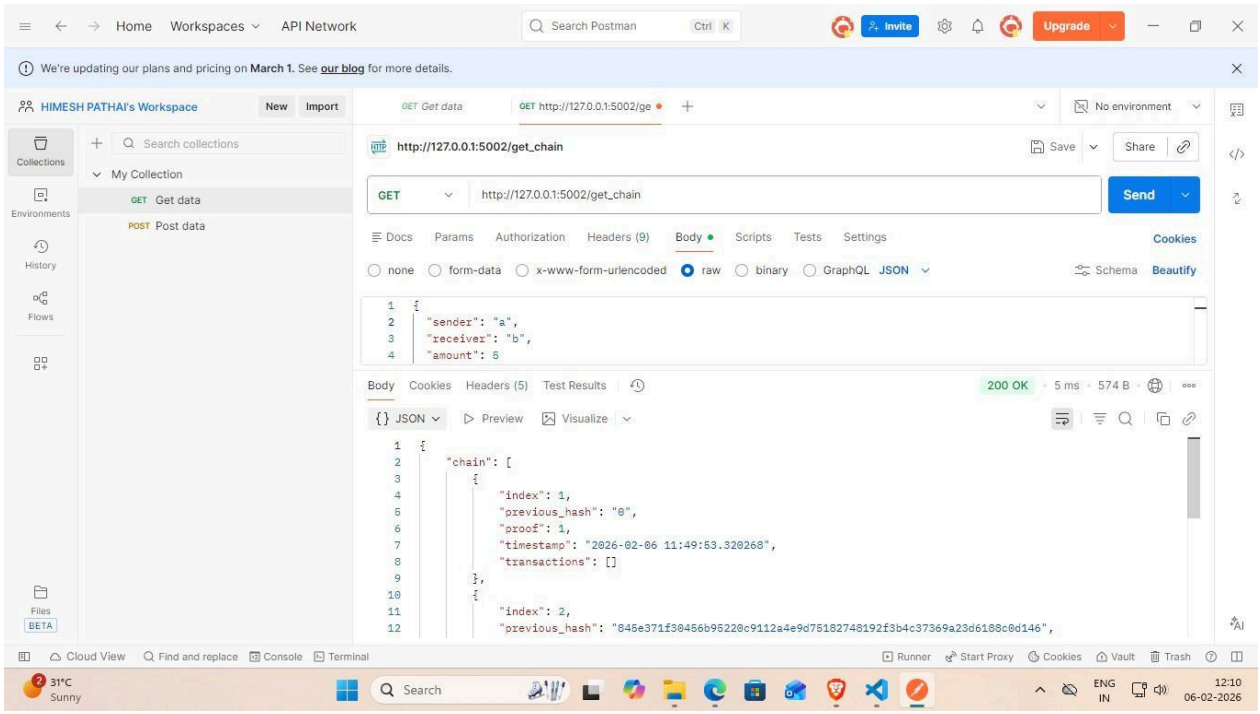


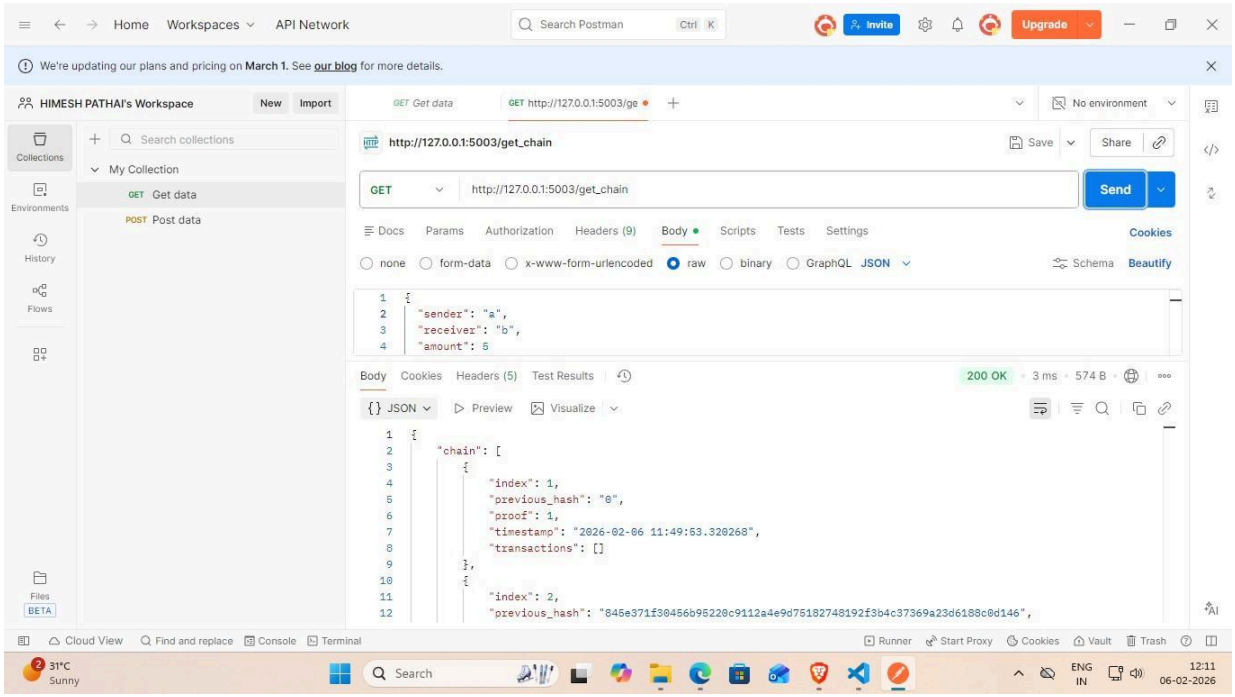
9. Mine blocks – GET `/mine_block`.



10. Fetch blockchain – GET `/get_chain`.




Home
Workspaces
API Network
Search Postman
Invite
Upgrade
We're updating our plans and pricing on March 1. See our blog for more details.
HIMESH PATHAI's Workspace
New
Import
Collections
Environments
History
Flows
Files
BETA
Search collections
My Collection
GET Get data
POST Post data
GET http://127.0.0.1:5002/ge
No environment
http://127.0.0.1:5002/get_chain
Save
Share
GET
http://127.0.0.1:5002/get_chain
Send
Docs
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{
"sender": "a",
"receiver": "b",
"amount": 5
}
Body
Cookies
Headers (5)
Test Results
200 OK
5 ms
574 B
{} JSON
Preview
Visualize
{
"chain": [
{
"index": 1,
"previous_hash": "0",
"proof": 1,
"timestamp": "2026-02-06 11:49:53.320268",
"transactions": []
},
{
"index": 2,
"previous_hash": "845e371f30456b95220c9112a4e9d75182748192f3b4c37369a23d6188c0d146",
Cloud View
Find and replace
Console
Terminal
Runner
Start Proxy
Cookies
Vault
Trash
31°C
Sunny
Search
ENG
IN
12:10
06-02-2026


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
We're updating our plans and pricing on March 1. See our blog for more details.
HIMESH PATHAI's Workspace
New
Import
Collections
Environments
History
Flows
Files
BETA
Search collections
My Collection
GET Get data
POST Post data
GET http://127.0.0.1:5003/ge
No environment
http://127.0.0.1:5003/get_chain
Save
Share
GET
http://127.0.0.1:5003/get_chain
Send
Docs
Params
Authorization
Headers (9)
Body
Scripts
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Schema
Beautify
{
"sender": "a",
"receiver": "b",
"amount": 5
}
Body
Cookies
Headers (5)
Test Results
200 OK
3 ms
574 B
{} JSON
Preview
Visualize
{
"chain": [
{
"index": 1,
"previous_hash": "0",
"proof": 1,
"timestamp": "2026-02-06 11:49:53.320268",
"transactions": []
},
{
"index": 2,
"previous_hash": "845e371f30456b95220c9112a4e9d75182748192f3b4c37369a23d6188c0d146",
Cloud View
Find and replace
Console
Terminal
Runner
Start Proxy
Cookies
Vault
Trash
31°C
Sunny
Search
ENG
IN
12:11
06-02-2026

Conclusion:

We developed a cryptocurrency using Python and implemented mining in a simulated three-node blockchain network. Each node maintained its own ledger and synchronized with peers using the Longest Chain Rule to ensure consistency. Mining was performed through Proof-of-Work, securely adding transactions and rewarding miners. This demonstrated key blockchain concepts, including decentralized consensus, transaction validation, and network synchronization.