BlockchainProject-Dexter's CoffeeShop

GroupNo.18

MEMBERS

YASHANK GARG : 2019A7PS0347H

SHIVAM AGRAWAL : 2019AAPS0326H

DEVASHISH GIDWANI : 2019A3PS0389H

Requirements to be installed:

- Flask==0.12.2:pipinstallFlask==0.12.2
- Postman HTTP Client: https://www.getpostman.com/
- requests==2.18.4:pipinstallrequests==2.18.4
- Pip install p2pnetwork

ProgrammingLanguageUsed: Python(Version:3.9)

Proof of elapsed time (PoET):

❖ A consensus algorithm developed by Intel Corporation that enables permissioned blockchain networks to determine block winners and mining rights.

❖ The **trusted execution environment** (**TEE**) in the network is achieved by Intel's **Software Guard Extensions** (**SGX**), which are instruction sets that allow user code to allocate private memory regions.

❖ PoET follows a lottery system that spreads the chances of winning equally across network participants, giving every single node the same chance of winning.

❖ The PoET algorithm generates a random wait time for each node in the blockchain network; each node must sleep for that duration.

❖ The node with the shortest wait time will wake up first and win the block, thus being allowed to commit a new block to the blockchain.

❖ The PoET workflow is similar to Bitcoin's proof of work (PoW) but consumes less power because it allows a miner's processor to sleep and switch to other tasks for the specified time, thereby increasing efficiency.

THE ACTUAL IMPLEMENTATION BY INTEL:

- ❖ POeT is based on special CPUs developed by Intel called SGX — Software Guard Extensions. SGX allows a logical separation of the CPU memory that cannot be accessed or changed. These parts are also called enclaves and can perform isolated commands and memory encryption. The code is encrypted and cannot be accessed outside the enclave, making it very secure for the processes that happen inside the enclave.
- ❖ To participate as a node in Sawtooth's POeT consensus, the node needs to download the application software and gain a membership certificate. The code will then generate a key pair for the participating node. SGX uses an asymmetric key approach with the private/public key pair. The miner needs to send SGX's attestation to the network in order to be approved.

OUR IMPLEMENTATION:

- ❖ Each transaction is initially added to a common transaction pool before the miner selects them to include in the block. After the transaction has been added to the mined block, it is removed from the transaction pool.
- ❖ Every participant in the network is assigned a random amount of time to wait, and the first participant to finish waiting gets to commit the next block to the blockchain.
- ❖ This specific functionality has been implemented in the file named **POET.py**

A Block contains :

- Block
- Previous Hash
- Timestamp
- Transactions
- Block Index
- Miner Details (the node which had the least wait-time and mined the block)

A transaction includes:

- Customer
- Receiver
- Amount of order
- Order details
- Order DateTime

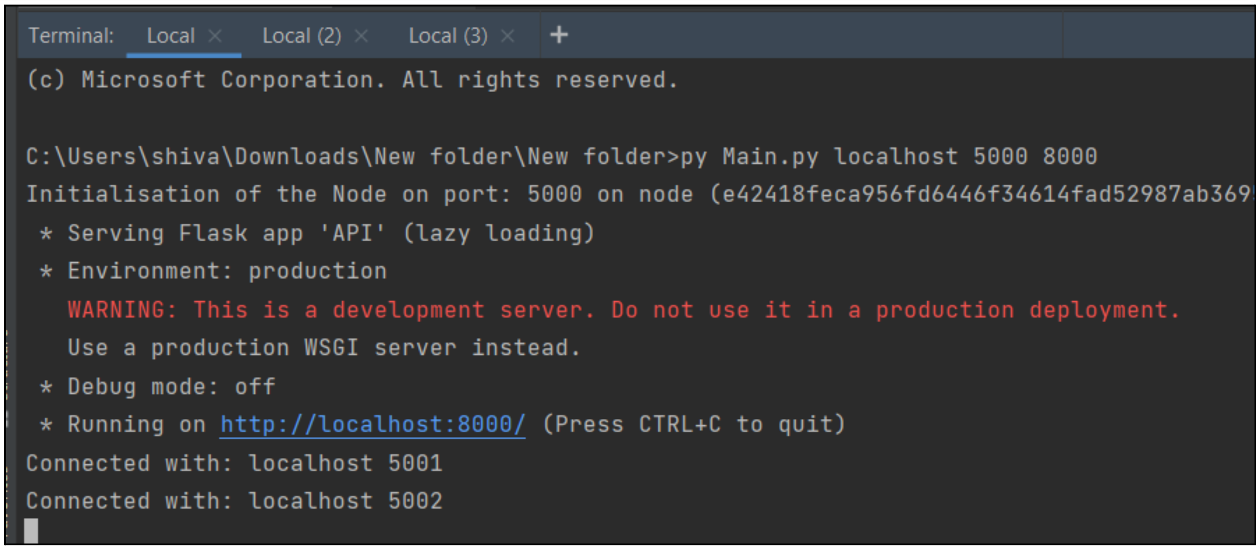
ESTABLISHING PEER-TO-PEER DISTRIBUTED NETWORK:

- ❖ The library **p2pnetwork 1.1** provides a basic and simple peer-to-peer decentralized network classes (framework) to build our own network.
- ❖ LINK: https://pypi.org/project/p2pnetwork/
- ❖ We Implemented p2p application by extending Node and NodeConnection classes provided. Below are some of the methods used:

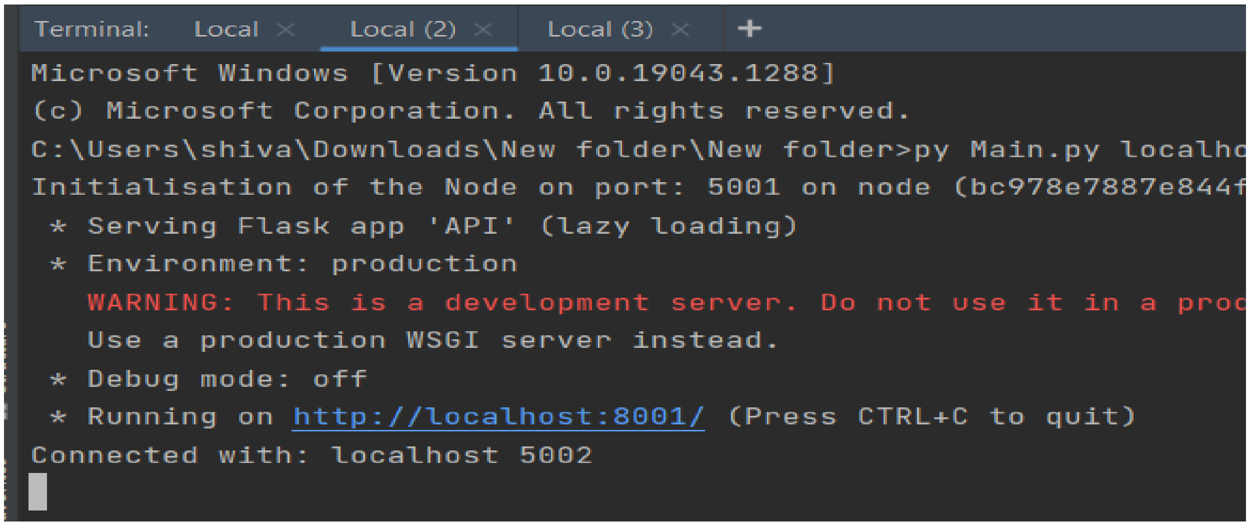
```
    def outbound_node_connected


    def inbound_node_connected


    def inbound_node_disconnected


    def outbound_node_disconnected


    def node_message
```

Adding and connecting nodes -

The first node in the network is at port 5000. When a new node joins, it connects to the node at 5000 to obtain the list of all nodes in the network. It now connects to other nodes in the network.

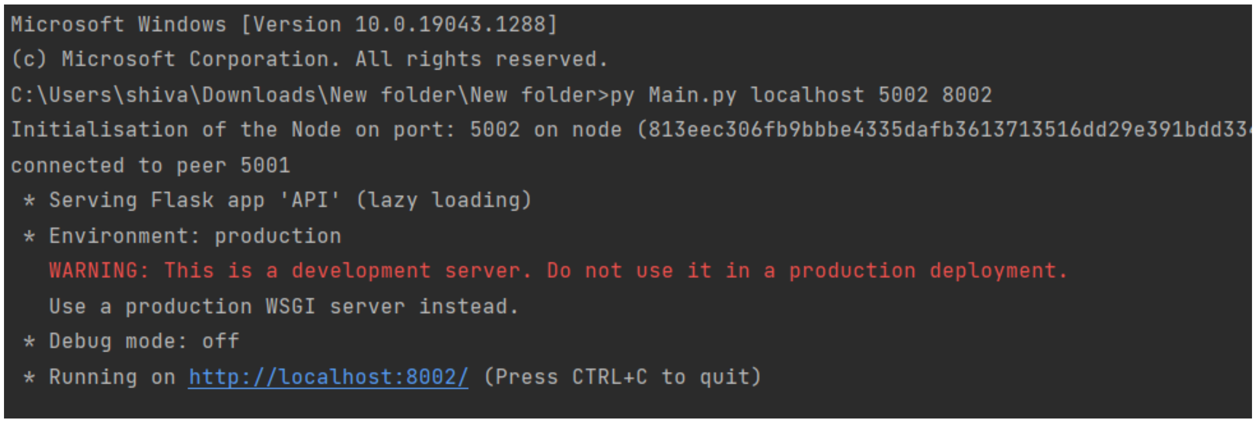


```
(c) Microsoft Corporation. All rights reserved.

C:\Users\shiva\Downloads\New folder\New folder>py Main.py localhost 5000 8000
Initialisation of the Node on port: 5000 on node (e42418feca956fd6446f34614fad52987ab369
 * Serving Flask app 'API' (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 http://localhost:8000/ (Press CTRL+C to quit)
Connected with: localhost 5001
Connected with: localhost 5002
```



```
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.
C:\Users\shiva\Downloads\New folder\New folder>py Main.py localho
Initialisation of the Node on port: 5001 on node (bc978e7887e844f
 * Serving Flask app 'API' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a proc
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://localhost:8001/ (Press CTRL+C to quit)
Connected with: localhost 5002
```

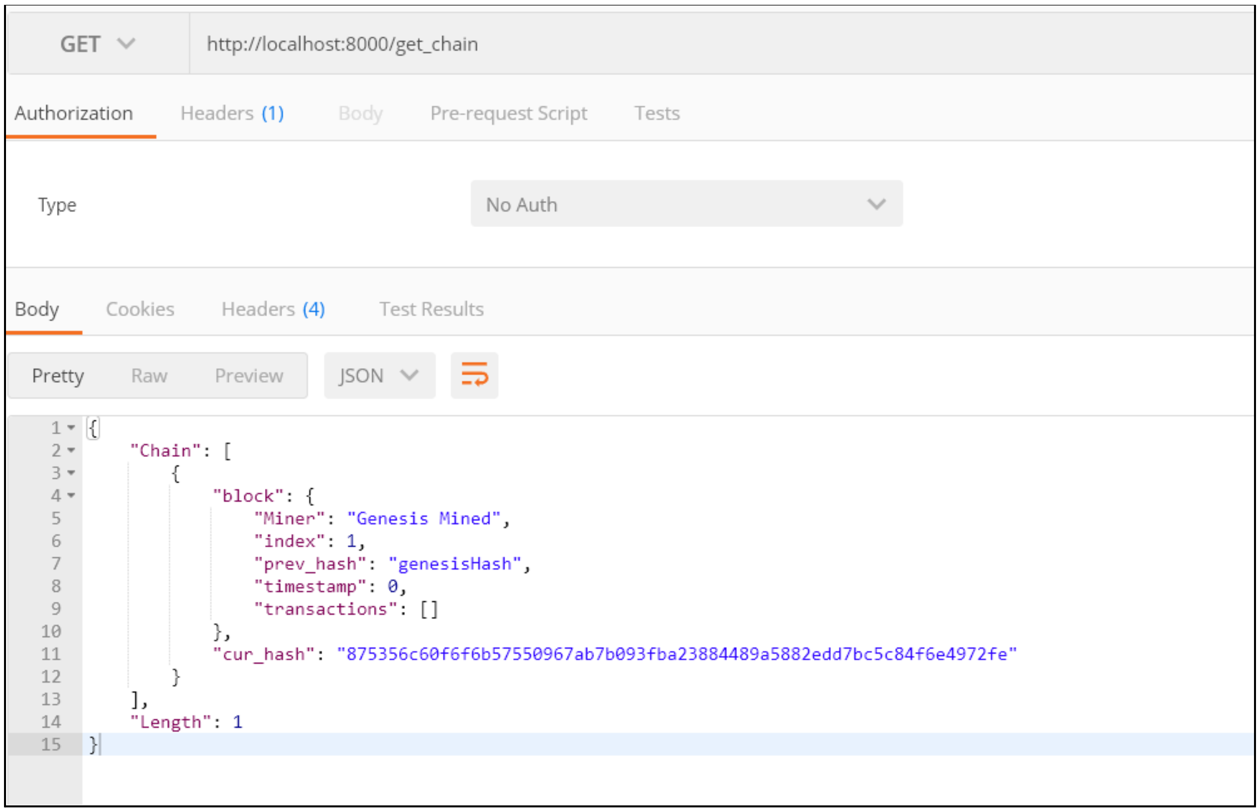
Here localhost 5002, entered the network in the end. When it connected to the other 2 nodes, both showed a connection message.



```
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.
C:\Users\shiva\Downloads\New folder\New folder>py Main.py localhost 5002 8002
Initialisation of the Node on port: 5002 on node (813eec306fb9bbbe4335dafb3613713516dd29e391bdd33
connected to peer 5001
 * Serving Flask app 'API' (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 http://localhost:8002/ (Press CTRL+C to quit)
```

Obtaining the details of the chain using REST API

The blockchain is initialized with the genesis block just as a node enters the network.



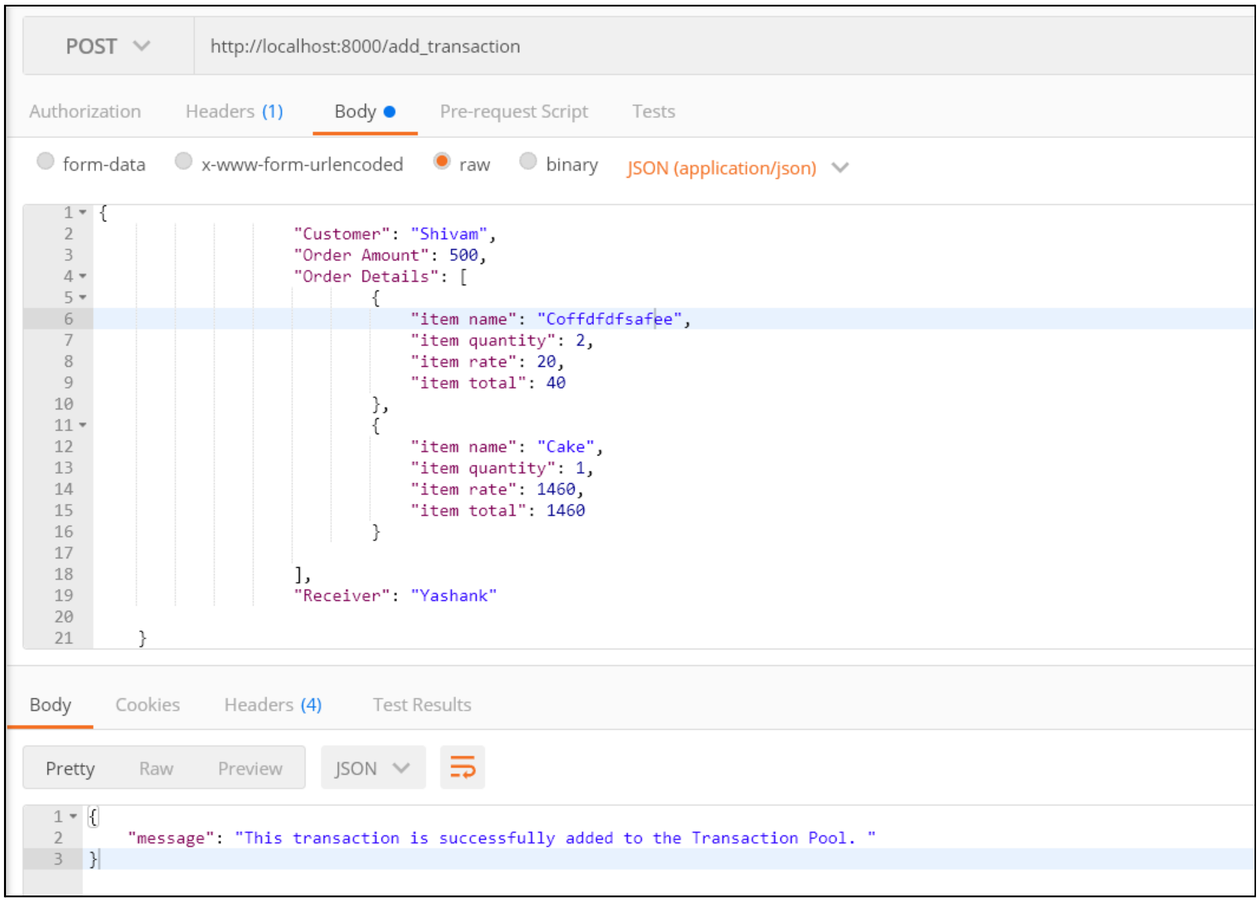
GET http://localhost:8000/get_chain

Type: No Auth

```
{
    "Chain": [
        {
            "block": {
                "Miner": "Genesis Mined",
                "index": 1,
                "prev_hash": "genesisHash",
                "timestamp": 0,
                "transactions": []
            },
            "cur_hash": "875356c60f6f6b57550967ab7b093fba23884489a5882edd7bc5c84f6e4972fe"
        }
    ],
    "Length": 1
}
```

Adding a new transaction -

To add a new transaction, a user can use the **/add_transaction** [POST request] in the Flask App.
The transaction is added to a common transaction pool shared by all nodes.
The transactions are added to a block and mined once they reach a certain limit. Here the limit is 1 transaction per block (can be changed in the code) for simplification purposes.
Once a transaction becomes a part of a mined block, it is removed from the transaction pool.



```
http://localhost:8000/add_transaction
```

```
{
                "Customer": "Shivam",
                "Order Amount": 500,
                "Order Details": [
                        {
                            "item name": "Coffdfdfsafee",
                            "item quantity": 2,
                            "item rate": 20,
                            "item total": 40
                        },
                        {
                            "item name": "Cake",
                            "item quantity": 1,
                            "item rate": 1460,
                            "item total": 1460
                        }

                ],
                "Receiver": "Yashank"

    }
```

```
{
    "message": "This transaction is successfully added to the Transaction Pool. "
}
```

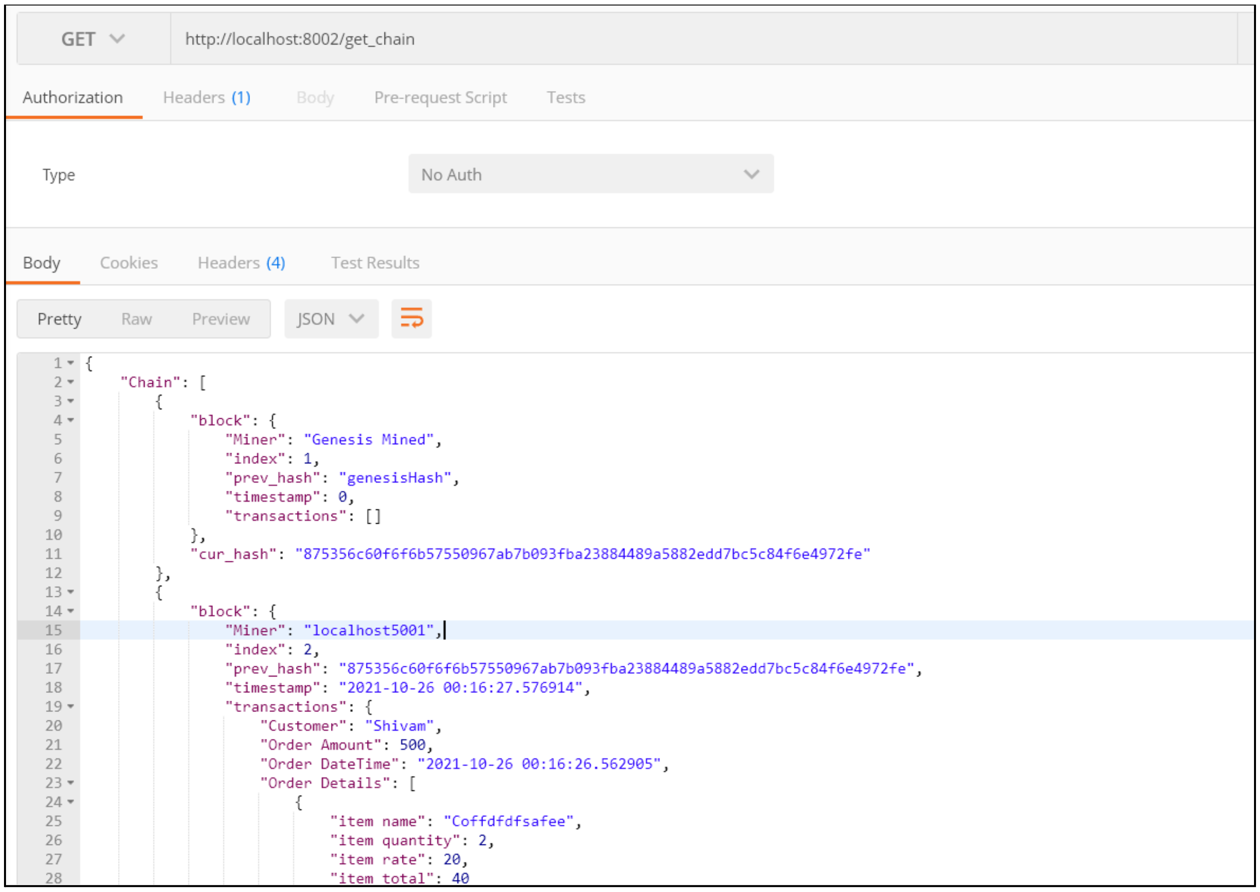
Choosing the miner -

When it is time to mine the new block, a timer is set with random wait time (between 1 and 20 seconds) for each node. When a node finishes its waiting time, a signal is sent to other nodes to stop waiting. The node who has completed the time first wins. It creates a block and sends it to other nodes. The blockchain is updated in each node.

Below we can see the representation of the current status of the chain after adding the above-shown transaction.The transaction was added by node at port 5000 (api port 8000) . As we can see, each node has a copy of the block with the same miner (port 5001), showing that this block was mined by the node at port 5001

Viewing The Chain: /get_chain [GET request]

To view the latest version of the blockchain, any node can call the above get request and receive a detailed view of the entire blockchain.
Below we observe the nodes at API ports 8000,8001, 8002 calling the get_chain separately and receiving the same blockchain.

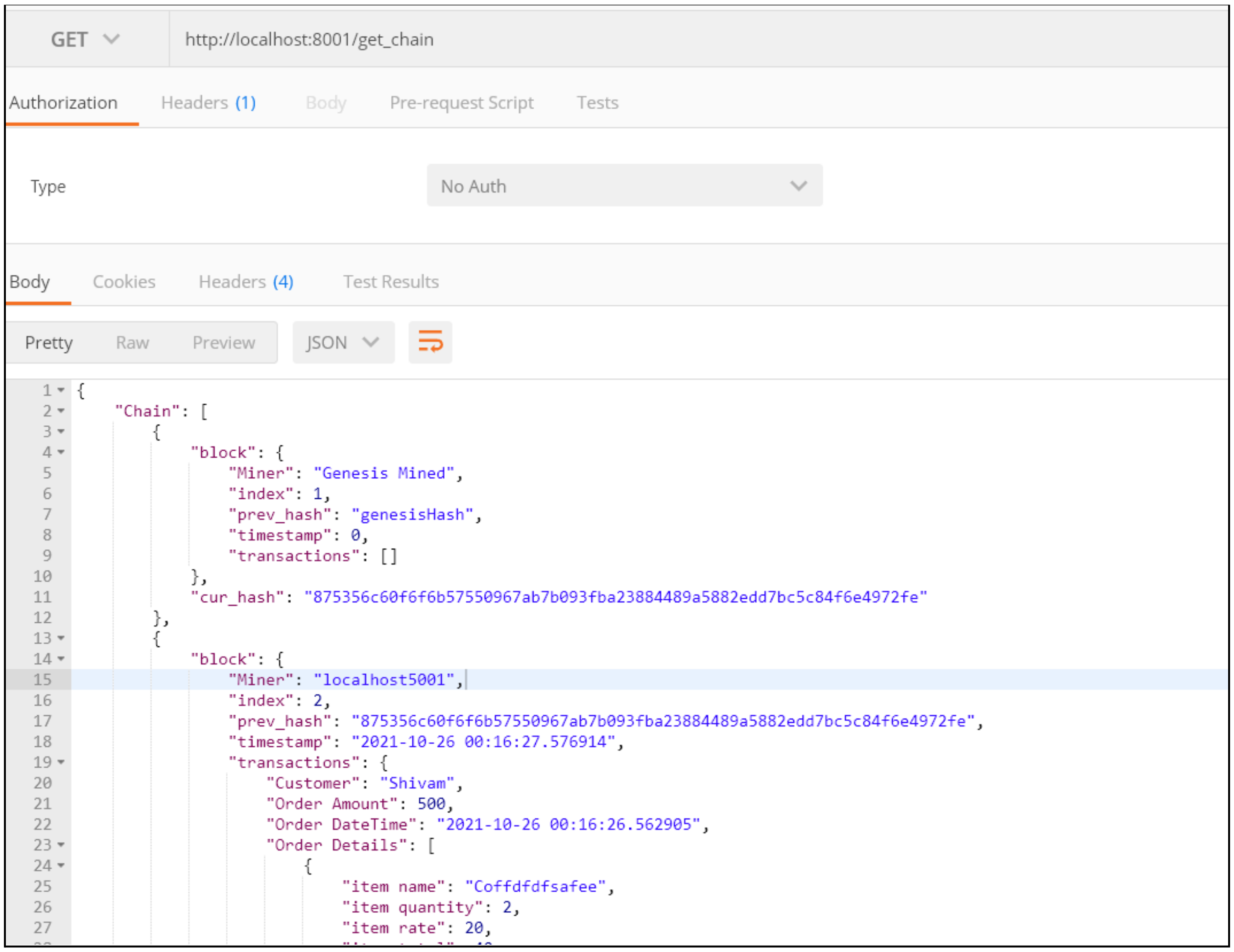


http://localhost:8002/get_chain

Type: No Auth

```
{
    "Chain": [
        {
            "block": {
                "Miner": "Genesis Mined",
                "index": 1,
                "prev_hash": "genesisHash",
                "timestamp": 0,
                "transactions": []
            },
            "cur_hash": "875356c60f6f6b57550967ab7b093fba23884489a5882edd7bc5c84f6e4972fe"
        },
        {
            "block": {
                "Miner": "localhost5001",
                "index": 2,
                "prev_hash": "875356c60f6f6b57550967ab7b093fba23884489a5882edd7bc5c84f6e4972fe",
                "timestamp": "2021-10-26 00:16:27.576914",
                "transactions": {
                    "Customer": "Shivam",
                    "Order Amount": 500,
                    "Order DateTime": "2021-10-26 00:16:26.562905",
                    "Order Details": [
                        {
                            "item name": "Coffdfdfsafee",
                            "item quantity": 2,
                            "item rate": 20,
                            "item total": 40
```

Port 8001 calling get_chain

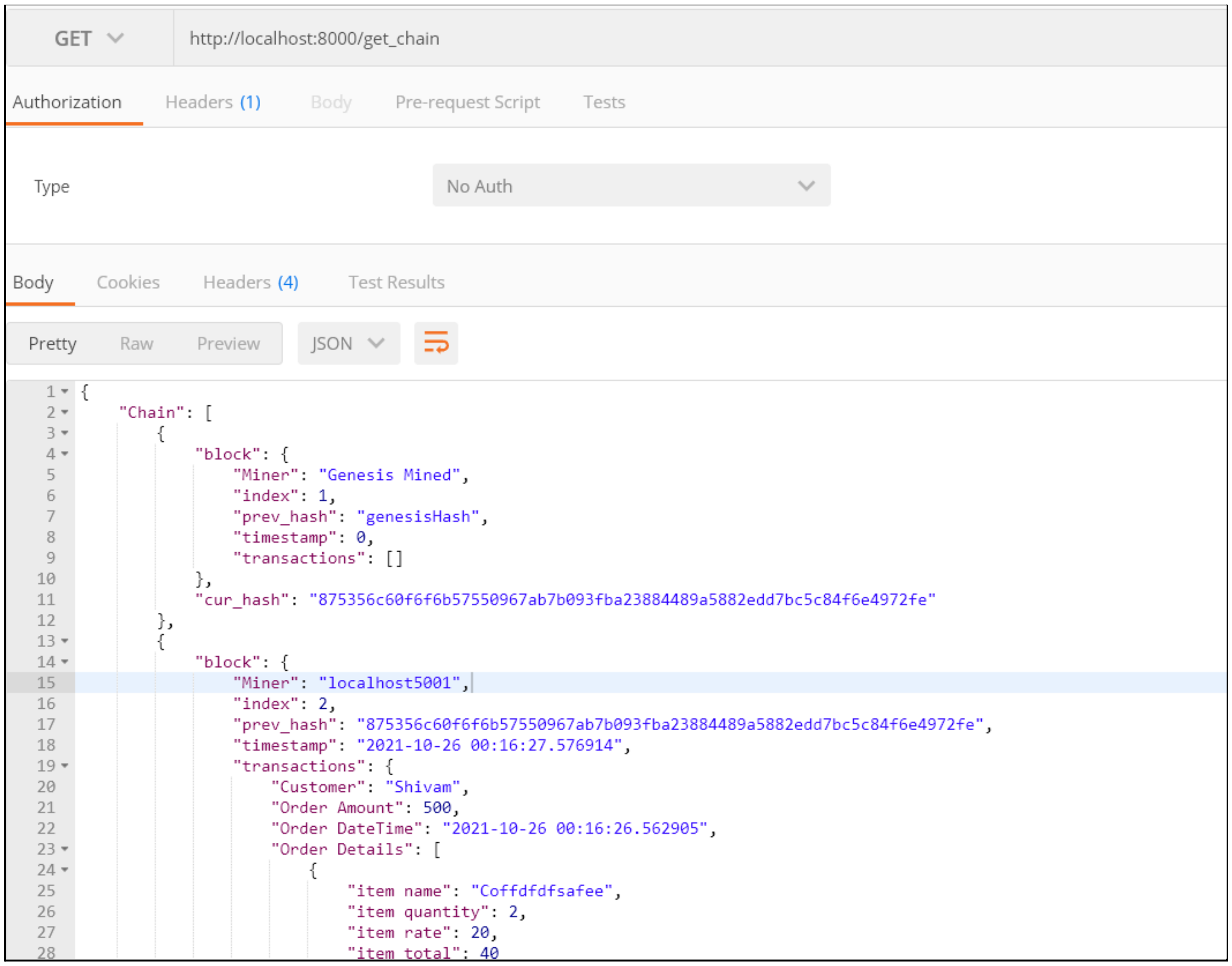


http://localhost:8001/get_chain

Type No Auth

```
{
    "Chain": [
        {
            "block": {
                "Miner": "Genesis Mined",
                "index": 1,
                "prev_hash": "genesisHash",
                "timestamp": 0,
                "transactions": []
            },
            "cur_hash": "875356c60f6f6b57550967ab7b093fba23884489a5882edd7bc5c84f6e4972fe"
        },
        {
            "block": {
                "Miner": "localhost5001",
                "index": 2,
                "prev_hash": "875356c60f6f6b57550967ab7b093fba23884489a5882edd7bc5c84f6e4972fe",
                "timestamp": "2021-10-26 00:16:27.576914",
                "transactions": {
                    "Customer": "Shivam",
                    "Order Amount": 500,
                    "Order DateTime": "2021-10-26 00:16:26.562905",
                    "Order Details": [
                        {
                            "item name": "Coffdfdfsafee",
                            "item quantity": 2,
                            "item rate": 20,
```

Port 8000 calling get_chain



GET http://localhost:8000/get_chain

Authorization | Headers (1) | Body | Pre-request Script | Tests

Type: No Auth

Body | Cookies | Headers (4) | Test Results

Pretty | Raw | Preview | JSON

```
{
    "Chain": [
        {
            "block": {
                "Miner": "Genesis Mined",
                "index": 1,
                "prev_hash": "genesisHash",
                "timestamp": 0,
                "transactions": []
            },
            "cur_hash": "875356c60f6f6b57550967ab7b093fba23884489a5882edd7bc5c84f6e4972fe"
        },
        {
            "block": {
                "Miner": "localhost5001",
                "index": 2,
                "prev_hash": "875356c60f6f6b57550967ab7b093fba23884489a5882edd7bc5c84f6e4972fe",
                "timestamp": "2021-10-26 00:16:27.576914",
                "transactions": {
                    "Customer": "Shivam",
                    "Order Amount": 500,
                    "Order DateTime": "2021-10-26 00:16:26.562905",
                    "Order Details": [
                        {
                            "item name": "Coffdfdfsafee",
                            "item quantity": 2,
                            "item rate": 20,
                            "item total": 40
```