Blockchain Project - Dexter's Coffee Shop

Group No. 18

Github link: https://github.com/Samdroid27/CoffeeHouse

Files included : node0.py, node1.py, node2.py (Nodes at port 5000, 5001 and 5002 respectively)

Requirements 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

Programming Language Used: Python (Version: 3.9)

Working of the Blockchain

For demonstration purposes, we have created 3 nodes at ports 5000, 5001, 5002 to create a decentralized blockchain and used Postman for API calls and presenting the corresponding response.

A Block contains :

- Block Index
- Previous Hash
- Nonce
- Timestamp
- Transactions

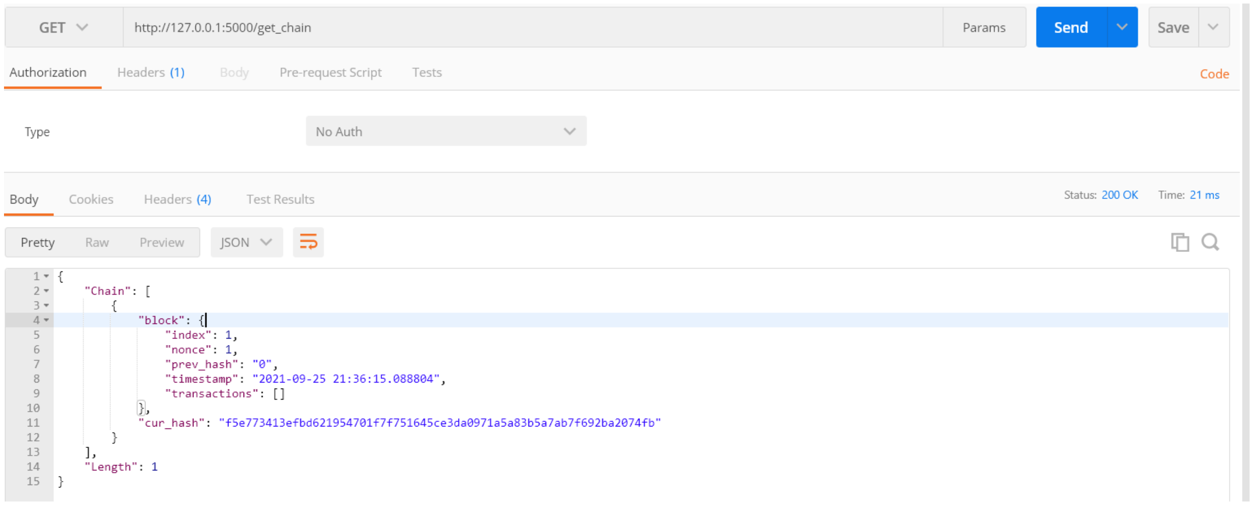
The difficulty level to mine a block :
A valid hash using SHA256 should have four leading zeros.

During verification, two constraints are checked for the validity of blockchain:

1. The previous hash matches the hash of the previous block
2. Hash of the current block has 4 leading zeros

Initializing the chain

Creation of the genesis block with block index 1.



http://127.0.0.1:5000/get_chain

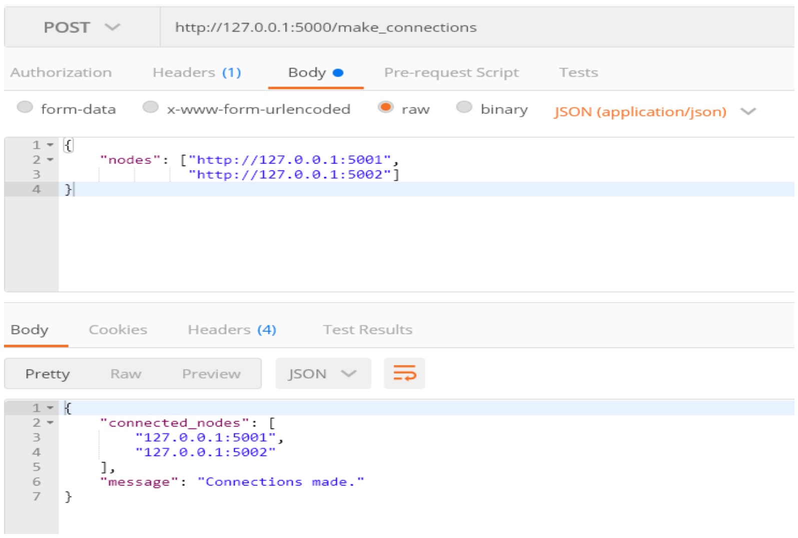
Type: No Auth

Status: 200 OK Time: 21 ms

```
{
    "Chain": [
        {
            "block": {
                "index": 1,
                "nonce": 1,
                "prev_hash": "0",
                "timestamp": "2021-09-25 21:36:15.088804",
                "transactions": []
            },
            "cur_hash": "f5e773413efbd621954701f7f751645ce3da0971a5a83b5a7ab7f692ba2074fb"
        }
    ],
    "Length": 1
}
```

Making connections between nodes - **[/make_connections , Request type= POST]**

Decentralization is an important property of blockchain. The node at one of the ports (5000, 5001, 5002) connects to the other nodes using the **make_connections()** function. One of the examples is shown below (node at port 5000 connecting to nodes at port 5001 and 5002).



http://127.0.0.1:5000/make_connections

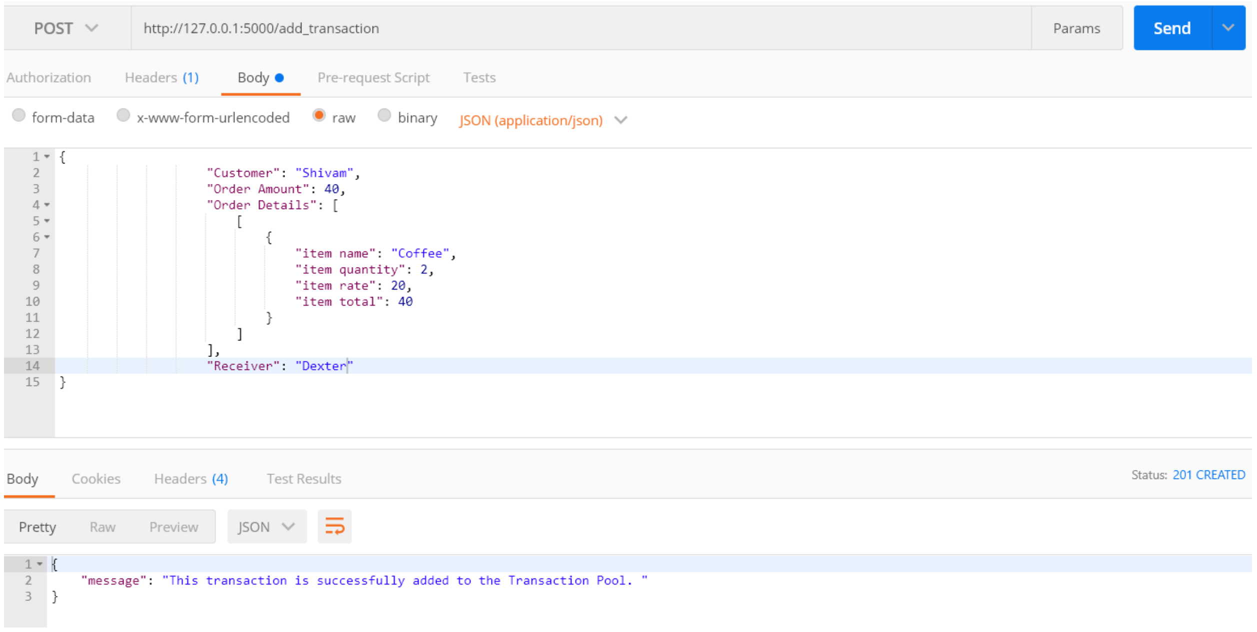
JSON (application/json)

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

```
{
    "connected_nodes": [
        "127.0.0.1:5001",
        "127.0.0.1:5002"
    ],
    "message": "Connections made."
}
```

Adding transaction - [/add_transaction , Request type= POST]

The transaction details are provided in a json file which includes customer name, order details, order amount, and the receiver. The timestamp of the order is auto-fetched and need not be provided explicitly.



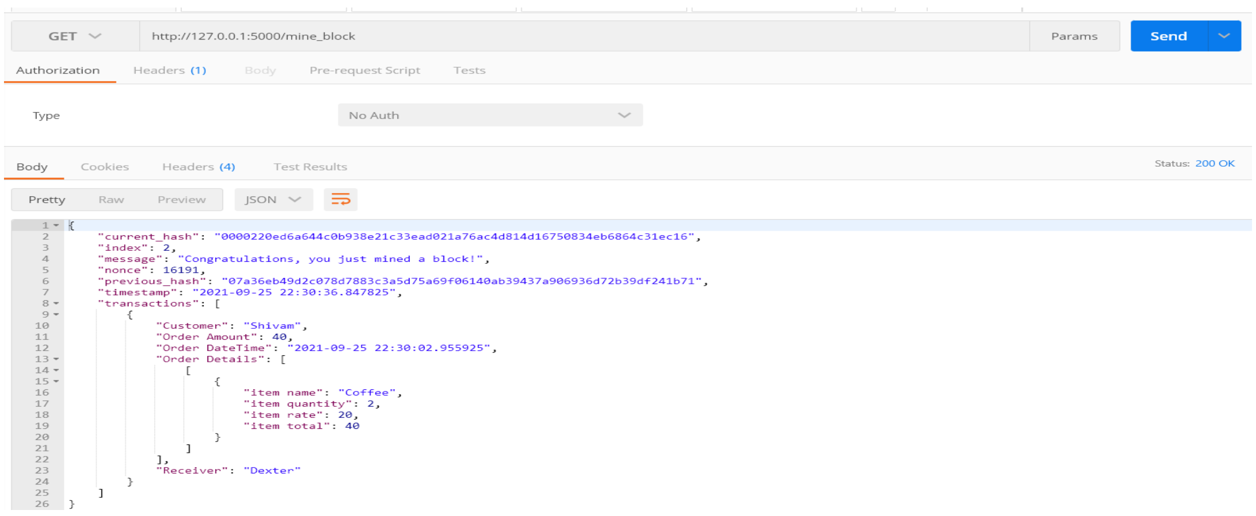
```
{
                "Customer": "Shivam",
                "Order Amount": 40,
                "Order Details": [
                    [
                        {
                            "item name": "Coffee",
                            "item quantity": 2,
                            "item rate": 20,
                            "item total": 40
                        }
                    ]
                ],
                "Receiver": "Dexter"
}
```

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

Mine A Block - - [/mine_block , Request type= GET]

To mine a block, a Node has to call **mine_block()**. This will first check if the current node has the latest blockchain or not, then accordingly add all the transactions currently in the transaction pool into a new block and append that block in the blockchain. The transaction pool is now empty. After mining, details of the newly created block are displayed.
We can see that the hash values of the blocks begin from 0000, which acts as proof of work. A suitable value of nonce is generated by brute force to achieve the same.

First, a block was mined on a node at port 5000. The chain is now of length 2.



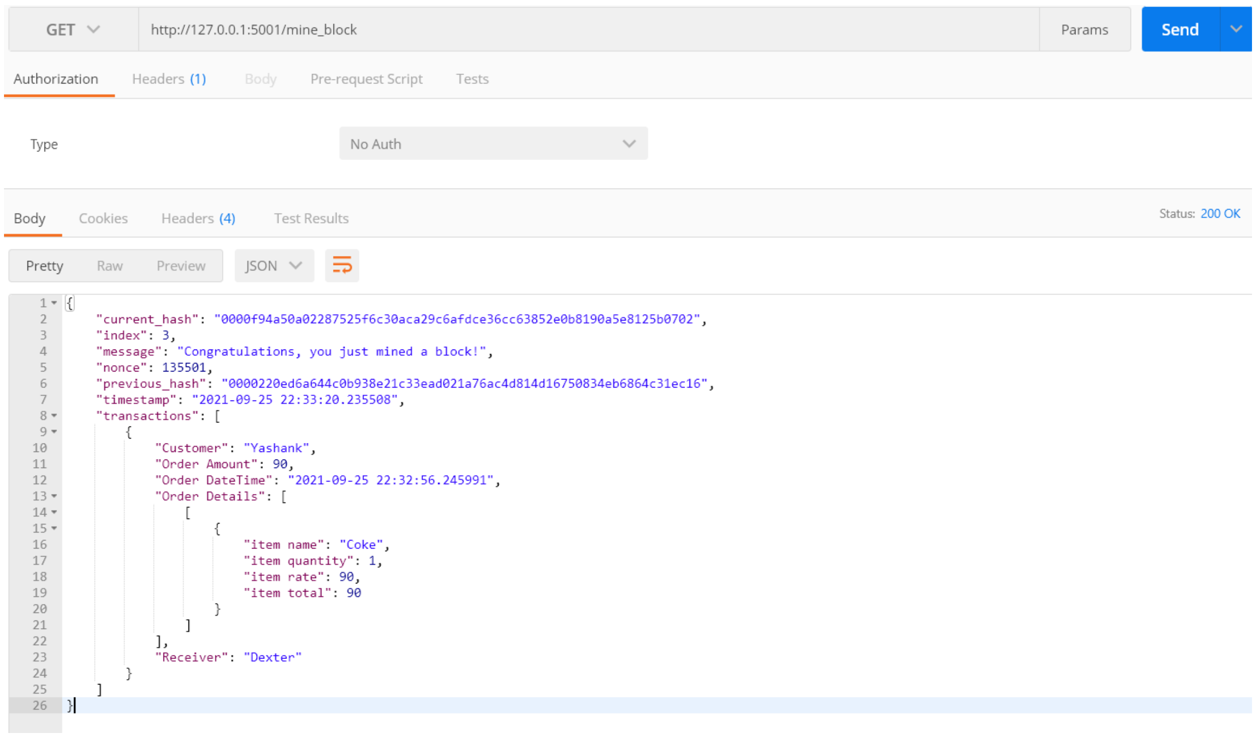
http://127.0.0.1:5000/mine_block

Type: No Auth

Status: 200 OK

```
{
    "current_hash": "0000220ed6a644c0b938e21c33ead021a76ac4d814d16750834eb6864c31ec16",
    "index": 2,
    "message": "Congratulations, you just mined a block!",
    "nonce": 16191,
    "previous_hash": "07a36eb49d2c078d7883c3a5d75a69f06140ab39437a906936d72b39df241b71",
    "timestamp": "2021-09-25 22:30:36.847825",
    "transactions": [
        {
            "Customer": "Shivam",
            "Order Amount": 40,
            "Order DateTime": "2021-09-25 22:30:02.955925",
            "Order Details": [
                [
                    {
                        "item name": "Coffee",
                        "item quantity": 2,
                        "item rate": 20,
                        "item total": 40
                    }
                ]
            ],
            "Receiver": "Dexter"
        }
    ]
}
```

After this, a block is mined on a node at port 5001. Before the block is added to the chain, it is checked whether the current chain is updated or not. Here we can see that the index of the block is 3, which means that the chain has been replicated from the node at port 5000 before adding the block. Now, this is the latest chain with length 3.



http://127.0.0.1:5001/mine_block

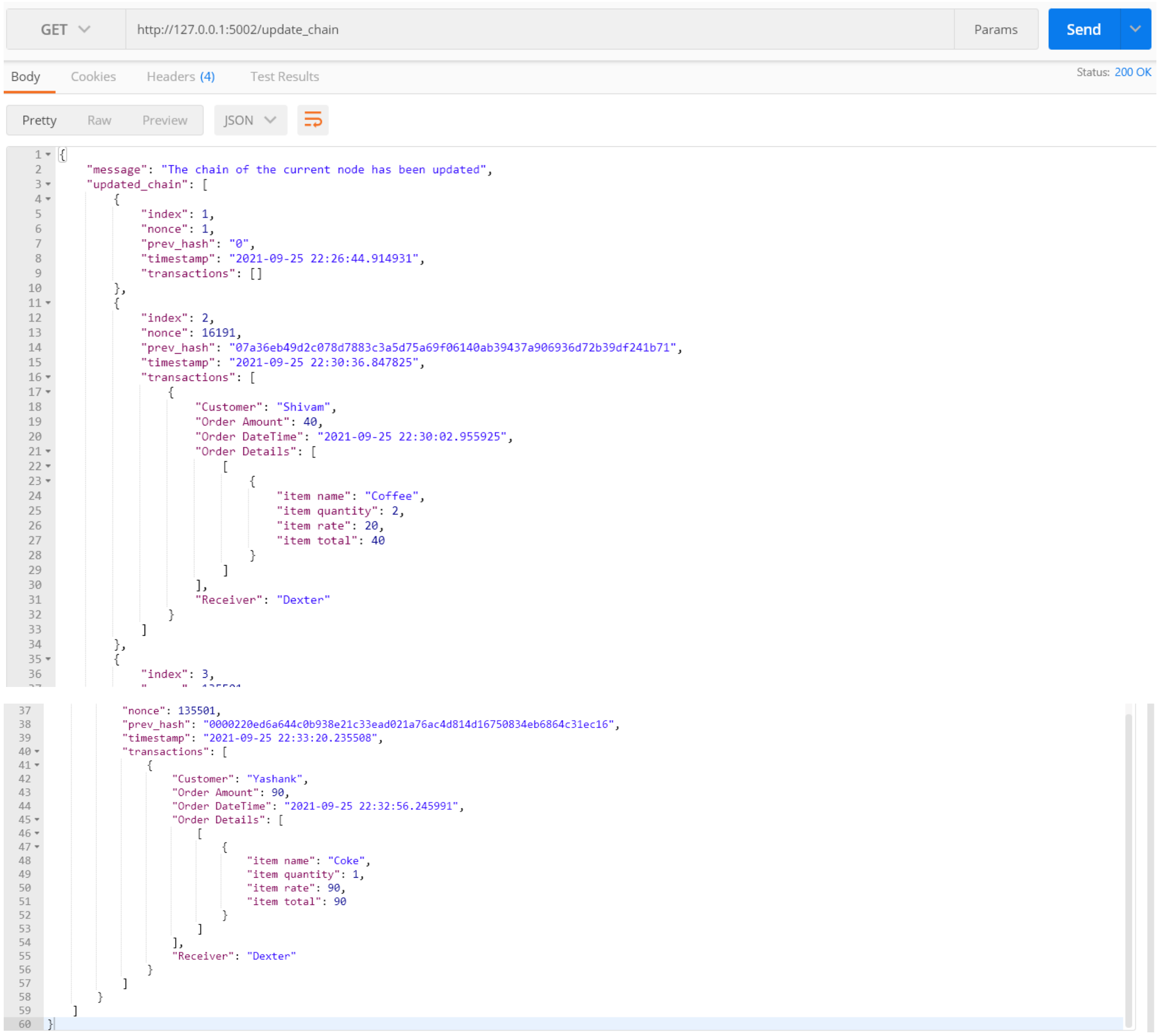
Type: No Auth

Status: 200 OK

```
{
    "current_hash": "0000f94a50a02287525f6c30aca29c6afdce36cc63852e0b8190a5e8125b0702",
    "index": 3,
    "message": "Congratulations, you just mined a block!",
    "nonce": 135501,
    "previous_hash": "0000220ed6a644c0b938e21c33ead021a76ac4d814d16750834eb6864c31ec16",
    "timestamp": "2021-09-25 22:33:20.235508",
    "transactions": [
        {
            "Customer": "Yashank",
            "Order Amount": 90,
            "Order DateTime": "2021-09-25 22:32:56.245991",
            "Order Details": [
                [
                    {
                        "item name": "Coke",
                        "item quantity": 1,
                        "item rate": 90,
                        "item total": 90
                    }
                ]
            ],
            "Receiver": "Dexter"
        }
    ]
}
```

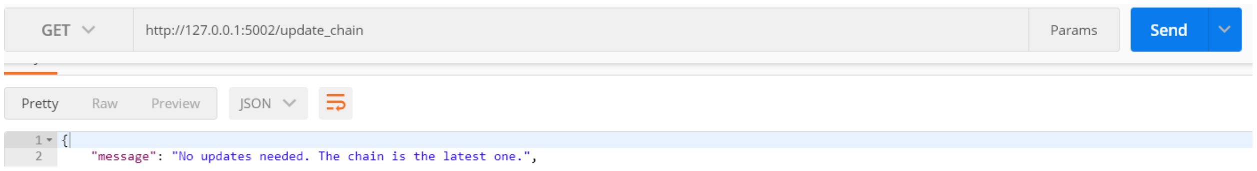
<u>Updating Chain</u> - [/update_chain , Request type= GET]

To check if the blockchain copy maintained by a node is the latest blockchain and to update it if it isn't, **update_chain()** is called. It replaces the chain of the current node with the latest chain across all nodes in the network. Here, the chain of the node at port 5002 is updated to the latest chain, present at the node at port 5001.



```
{
    "message": "The chain of the current node has been updated",
    "updated_chain": [
        {
            "index": 1,
            "nonce": 1,
            "prev_hash": "0",
            "timestamp": "2021-09-25 22:26:44.914931",
            "transactions": []
        },
        {
            "index": 2,
            "nonce": 16191,
            "prev_hash": "07a36eb49d2c078d7883c3a5d75a69f06140ab39437a906936d72b39df241b71",
            "timestamp": "2021-09-25 22:30:36.847825",
            "transactions": [
                {
                    "Customer": "Shivam",
                    "Order Amount": 40,
                    "Order DateTime": "2021-09-25 22:30:02.955925",
                    "Order Details": [
                        [
                            {
                                "item name": "Coffee",
                                "item quantity": 2,
                                "item rate": 20,
                                "item total": 40
                            }
                        ]
                    ],
                    "Receiver": "Dexter"
                }
            ]
        },
        {
            "index": 3,
            "nonce": 135501,
            "prev_hash": "0000220ed6a644c0b938e21c33ead021a76ac4d814d16750834eb6864c31ec16",
            "timestamp": "2021-09-25 22:33:20.235508",
            "transactions": [
                {
                    "Customer": "Yashank",
                    "Order Amount": 90,
                    "Order DateTime": "2021-09-25 22:32:56.245991",
                    "Order Details": [
                        [
                            {
                                "item name": "Coke",
                                "item quantity": 1,
                                "item rate": 90,
                                "item total": 90
                            }
                        ]
                    ],
                    "Receiver": "Dexter"
                }
            ]
        }
    ]
}
```

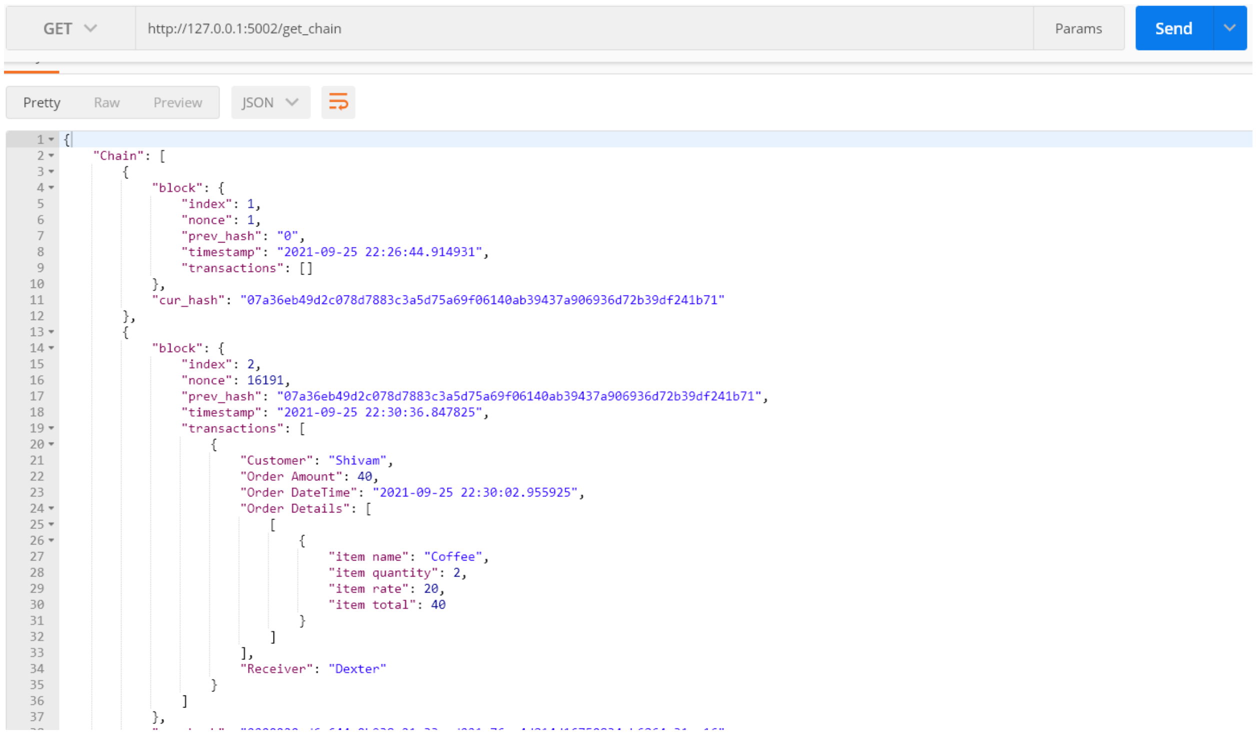
If the chain is already updated, below response is returned

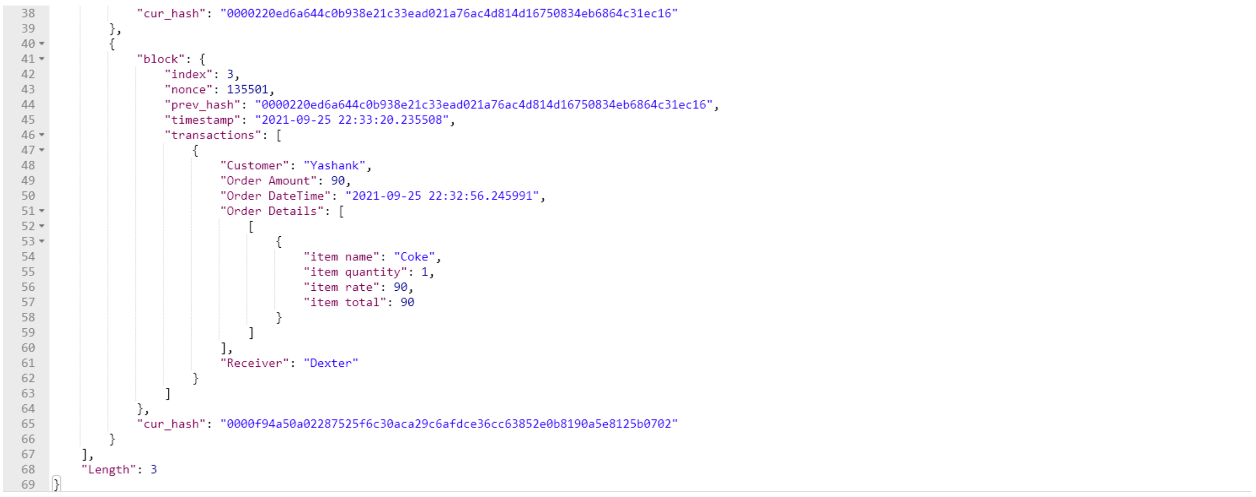


```
{
    "message": "No updates needed. The chain is the latest one.",
```

Get Blockchain Details [/get_chain , Request type= GET]

To get details of all the blocks in the blockchain, **get_chain()** is called. As we can observe, the prev_hash (previous hash) in the current block is the same as the hash of the previous block for all blocks, deeming the chain to be valid.

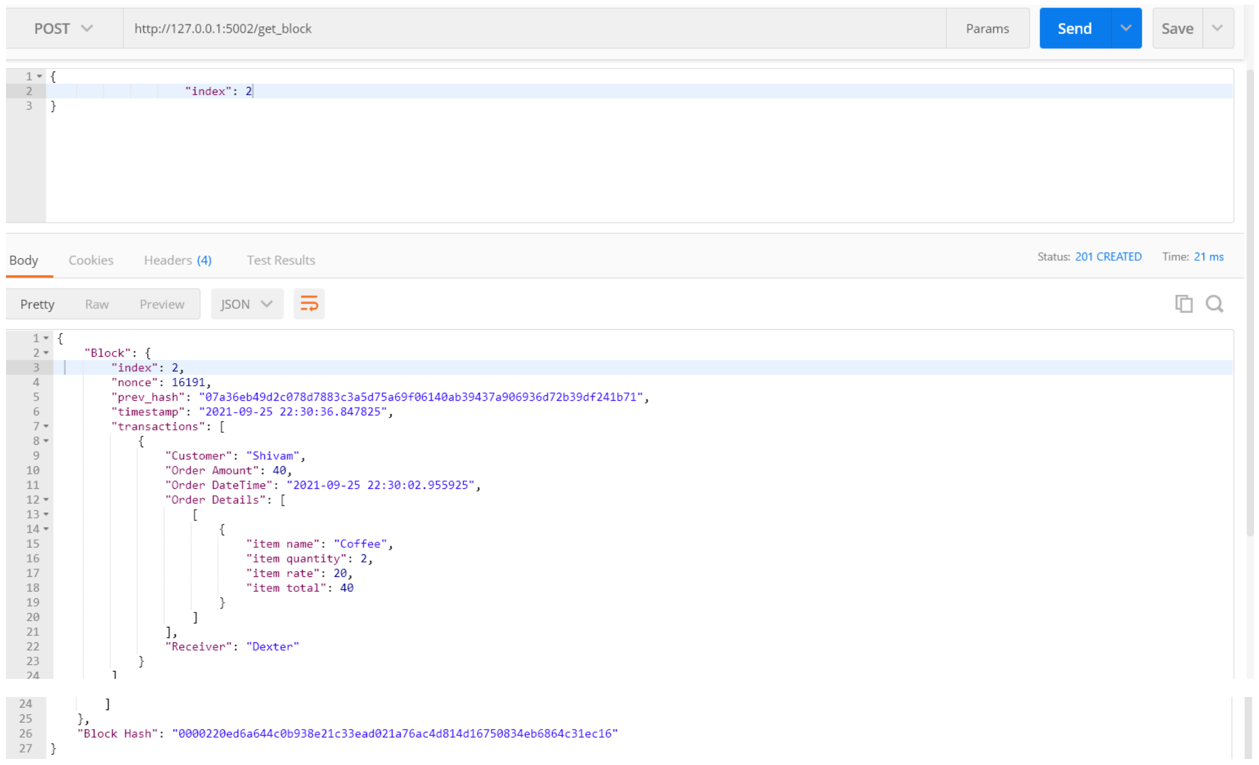




```
{
    "Chain": [
        {
            "block": {
                "index": 1,
                "nonce": 1,
                "prev_hash": "0",
                "timestamp": "2021-09-25 22:26:44.914931",
                "transactions": []
            },
            "cur_hash": "07a36eb49d2c078d7883c3a5d75a69f06140ab39437a906936d72b39df241b71"
        },
        {
            "block": {
                "index": 2,
                "nonce": 16191,
                "prev_hash": "07a36eb49d2c078d7883c3a5d75a69f06140ab39437a906936d72b39df241b71",
                "timestamp": "2021-09-25 22:30:36.847825",
                "transactions": [
                    {
                        "Customer": "Shivam",
                        "Order Amount": 40,
                        "Order DateTime": "2021-09-25 22:30:02.955925",
                        "Order Details": [
                            [
                                {
                                    "item name": "Coffee",
                                    "item quantity": 2,
                                    "item rate": 20,
                                    "item total": 40
                                }
                            ]
                        ],
                        "Receiver": "Dexter"
                    }
                ]
            },
            "cur_hash": "0000220ed6a644c0b938e21c33ead021a76ac4d814d16750834eb6864c31ec16"
        },
        {
            "block": {
                "index": 3,
                "nonce": 135501,
                "prev_hash": "0000220ed6a644c0b938e21c33ead021a76ac4d814d16750834eb6864c31ec16",
                "timestamp": "2021-09-25 22:33:20.235508",
                "transactions": [
                    {
                        "Customer": "Yashank",
                        "Order Amount": 90,
                        "Order DateTime": "2021-09-25 22:32:56.245991",
                        "Order Details": [
                            [
                                {
                                    "item name": "Coke",
                                    "item quantity": 1,
                                    "item rate": 90,
                                    "item total": 90
                                }
                            ]
                        ],
                        "Receiver": "Dexter"
                    }
                ]
            },
            "cur_hash": "0000f94a50a02287525f6c30aca29c6afdce36cc63852e0b8190a5e8125b0702"
        }
    ],
    "Length": 3
}
```

<u>Get Block Details</u> - [/get_block , Request type= POST]

To get Block Details of a particular block in the blockchain, Dexter has to provide the block_index of the desired block in json format.



POST http://127.0.0.1:5002/get_block

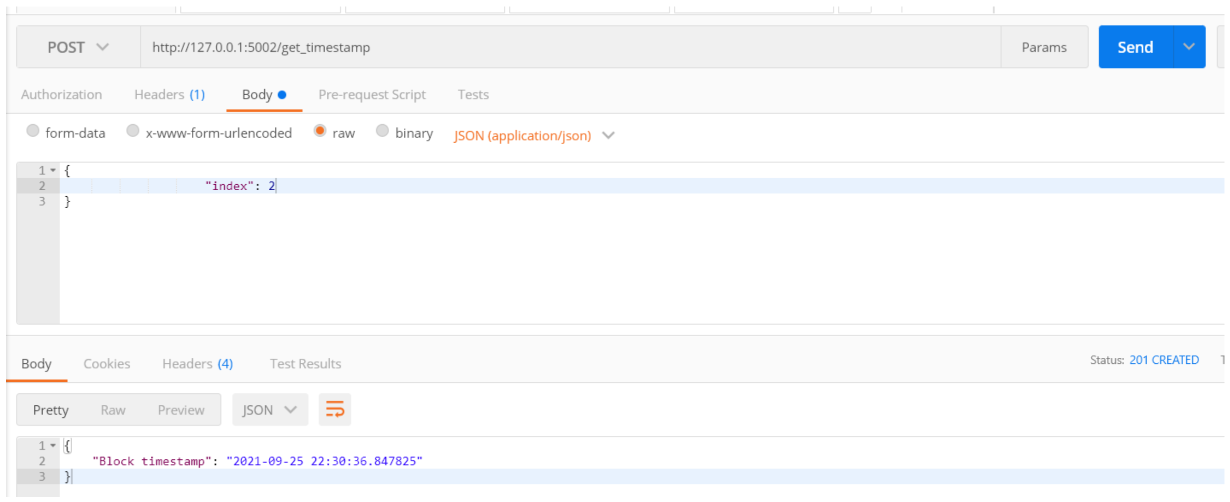
```
{
        "index": 2
}
```

Status: 201 CREATED Time: 21 ms

```
{
    "Block": {
        "index": 2,
        "nonce": 16191,
        "prev_hash": "07a36eb49d2c078d7883c3a5d75a69f06140ab39437a906936d72b39df241b71",
        "timestamp": "2021-09-25 22:30:36.847825",
        "transactions": [
            {
                "Customer": "Shivam",
                "Order Amount": 40,
                "Order DateTime": "2021-09-25 22:30:02.955925",
                "Order Details": [
                    [
                        {
                            "item name": "Coffee",
                            "item quantity": 2,
                            "item rate": 20,
                            "item total": 40
                        }
                    ]
                ],
                "Receiver": "Dexter"
            }
        ]
    },
    "Block Hash": "0000220ed6a644c0b938e21c33ead021a76ac4d814d16750834eb6864c31ec16"
}
```

<u>Get timestamp of specified block</u> - [/get_timestamp , Request type= POST]

To get the timestamp (time of creation) of a particular block in the blockchain, Dexter has to provide the block_index of the desired block in json format.



POST http://127.0.0.1:5002/get_timestamp

Body: raw, JSON (application/json)

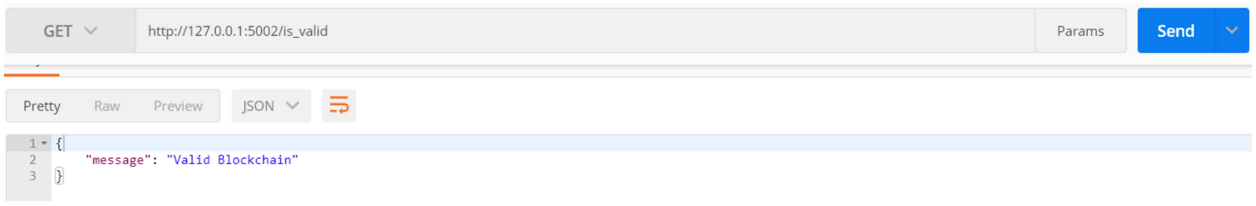
```
{
        "index": 2
}
```

Status: 201 CREATED

```
{
    "Block timestamp": "2021-09-25 22:30:36.847825"
}
```

<u>To check validity of the Blockchain</u> [/is_valid, Request type= GET]

For a node, to check if its copy of the blockchain follows all the constraints needed, **is_valid()** is called. It matches the previous hash mentioned in a block with the hash of the previous hash ,for every block in the chain.



http://127.0.0.1:5002/is_valid

```
{
    "message": "Valid Blockchain"
}
```