TERM PROJECT REPORT

CMSE 353
Security of Software Systems

Team members:

Roa’a Rami Abdel Ra’uof Alqaisi (Team Leader)	**19700652**
Dalal Totah	**19700331**
Mohammad Murra	**19700717**
Khawlah Al-Shubati	**19701557**

GROUP NO.: 01
Submitted to: Alexander Chefranov.

Computer Engineering Department Eastern Mediterranean University

Outline

Problem definition:

A cryptocurrency is a digital or virtual currency protected by encryption, making counterfeiting and double-spending practically impossible. Many cryptocurrencies are decentralized networks based on blockchain technology, which is a distributed ledger maintained by a network of computers. Cryptocurrencies are distinguished by the fact that they are not issued by a central authority, making them potentially impervious to government intervention or manipulation.

- The advantages of cryptocurrencies include cheaper and faster money transfers and decentralized systems that do not collapse at a single point of failure.
- The disadvantages of cryptocurrencies include their price volatility, high energy consumption for mining activities, and use in criminal activities.

In addition, Cryptocurrency enable secure online payments without the use of third-party intermediaries. "Crypto" refers to the various encryption algorithms and cryptographic techniques.

Teamwork description:

In our team we organized both online and face to face meetings so as to work cooperatively in the project. In addition to using online sources for gathering information about the topic we applied the necessary modifications and adjustments to our program. Most importantly, we divided the work among us in a way that each member of the group was in charge of a certain section.

Description of algorithm for solving the problem:

The solution we implemented of the problem we have is done using couple step by step well defined procedure that takes a set of values as input and gives us the result we want. As we know a blockchain is the world's most trusted service. It serves as a ledger that allows transaction to take place in a decentralized manner. In orde for the transaction to happen we need to enter data including (sender, recipient, amount transferred) and add more blocks along with more transactions happening.

In this procedure we will need to:

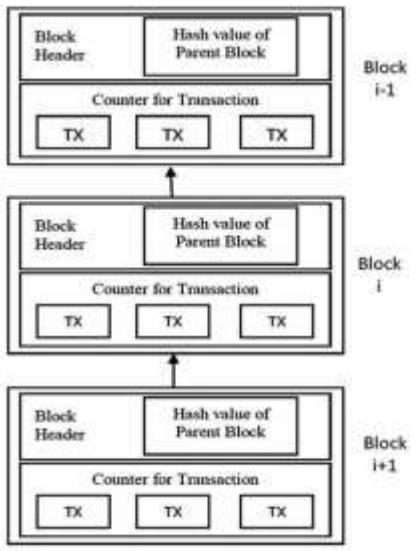
1. Build a blockchain (each blockchain has an index, a timestamp, a list of transactions, a proof of work, and the (hex №) hash of the previous block- for security purposes.

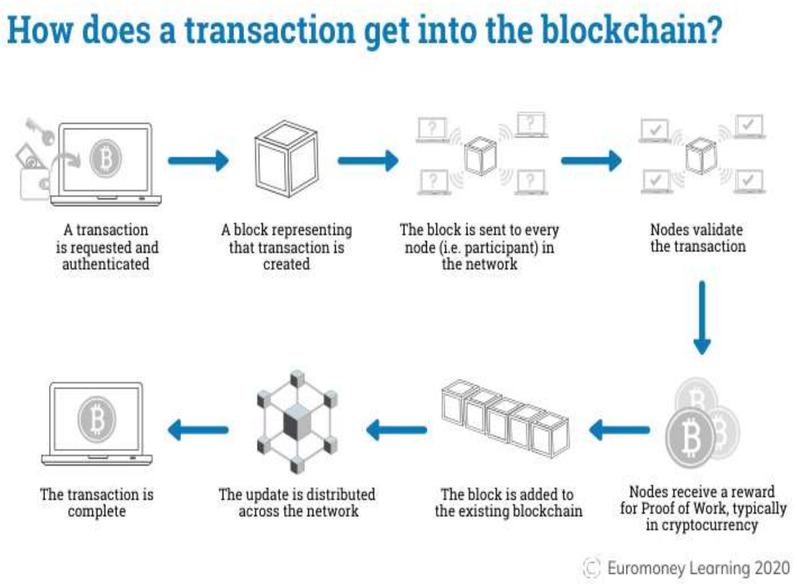
2. We will need to talk to our blockchain as an API (app programming interface) using HTTP requests using a micro-framework (the one we used to be the python flask framework).

3. Interact with our API over a network using cURL or Postman.(Developers use these to transfere data to and from a server)
   - Postman will be used as graphical user interface
   - cURL will be used as command line interface.

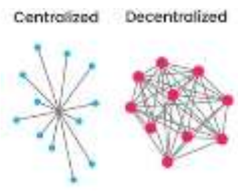
4. Consensus method: a. registering neighboring nodes.
   b. checking if a blockchain is valid. (if it has both the hash/proof)
   c. resolves any conflicts if more than one block chain, take the longer one.

We will also be using some libraries, and packages to import the needed extensions/functions needed. In addition to using transaction/mining endpoints. Not forgetting to mention the nodes need to be linked in a decentralized manner. (each participating node is independent of the others. Rather than following the instructions of a central authority connected using common rules).




How does a transaction get into the blockchain?
A transaction is requested and authenticated
A block representing that transaction is created
The block is sent to every node (i.e. participant) in the network
Nodes validate the transaction
Nodes receive a reward for Proof of Work, typically in cryptocurrency
The block is added to the existing blockchain
The update is distributed across the network
The transaction is complete
© Euromoney Learning 2020

Description of Database structure designed:



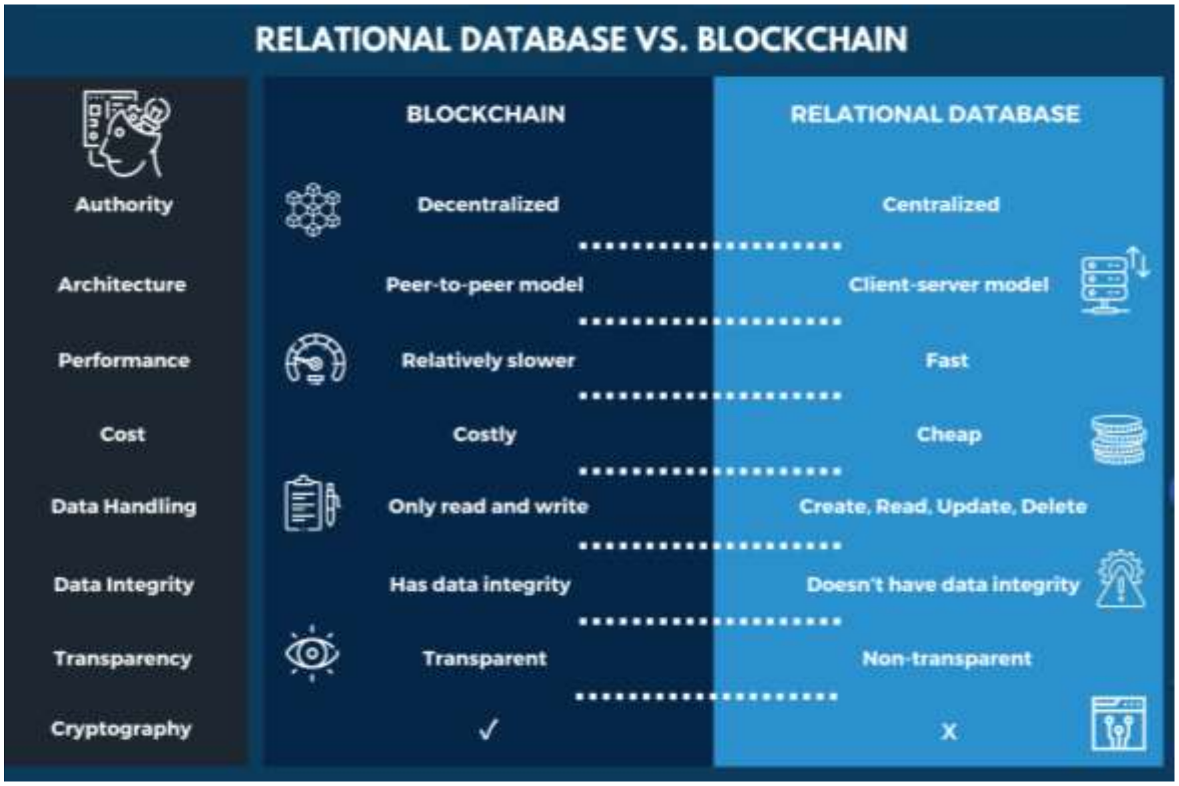
The blockchain is considered as data structure that stores transactions. It is similar to a linked list in which the data is split into blocks. Each block is connected with its predecessor with a cryptographically secured reference in which we call hash value.

A block consists of a header, and the transactions contained. Inside the block, a Merkle tree is used to create a 256-bit summary of all transactions, the Merkle root, which is included in the block header.

Spotting the difference between a blockchain and a relational database:

Databases are centralized ledger in which it stores data in a structured way and needs to be managed by an administrator in which only him can decide which the public can see. It is very easy and fast to implement since it was founded years before. Whereas a blockchain is peer to peer decentralized distributed ledger technology. No administrator needed and it offers transparency. It also supports verification and consensus methods.

Here is a briefly summarized table to show the difference:



RELATIONAL DATABASE VS. BLOCKCHAIN

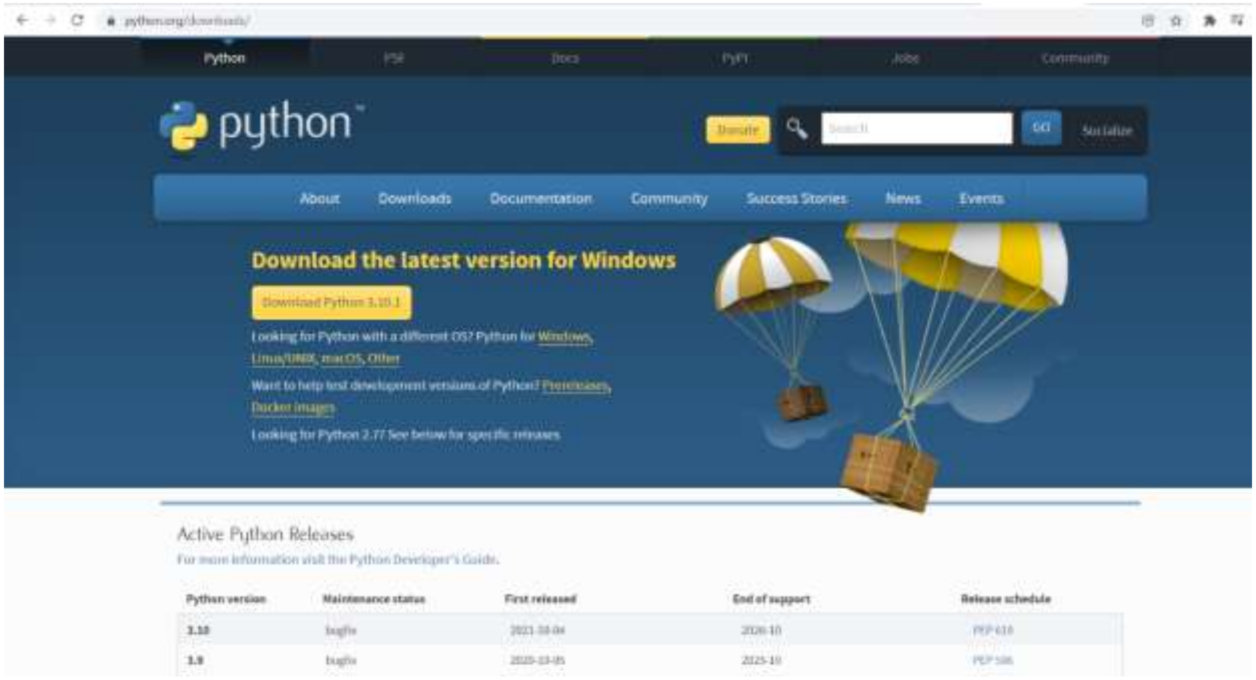
	BLOCKCHAIN	RELATIONAL DATABASE
Authority	Decentralized	Centralized
Architecture	Peer-to-peer model	Client-server model
Performance	Relatively slower	Fast
Cost	Costly	Cheap
Data Handling	Only read and write	Create, Read, Update, Delete
Data Integrity	Has data integrity	Doesn't have data integrity
Transparency	Transparent	Non-transparent
Cryptography	✓	X

Description of Tools and Data implementation:

We implemented our program using python programming language which by default offers some modules that were helpful when executing our code such as hashlib, json…etc. In addition to python, we needed Flask,and Requests library. For our application programming interface (API) we used postman. Accordingly, for any person wants to run and execute our program, they need to download the following:

1. Python.
2. Python IDE (optional)
3. Postman or any other API platform.

Steps to download python:



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

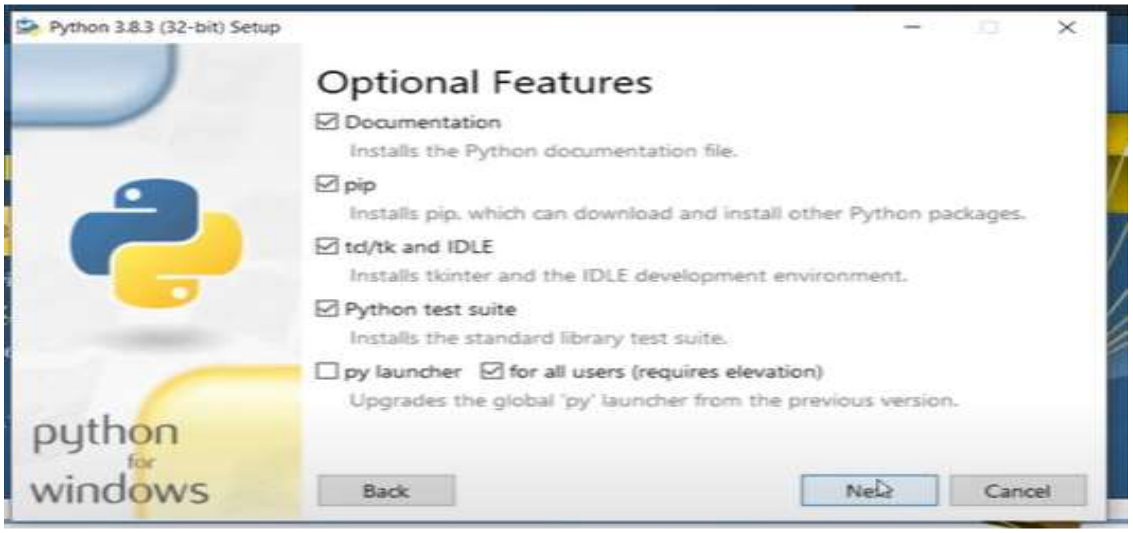
Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.10	bugfix	2021-10-04	2026-10	PEP 619
3.9	bugfix	2020-10-05	2025-10	PEP 596

First, you need to go to the official python website and go to the download section using the following link: https://www.python.org/downloads/

There are variety of options available depending on your Operating System that you are using. However, the website offers an automatic download button highlighted with yellow which download the latest version of python compatible with your operating system.



Optional Features

☑ Documentation
Installs the Python documentation file.

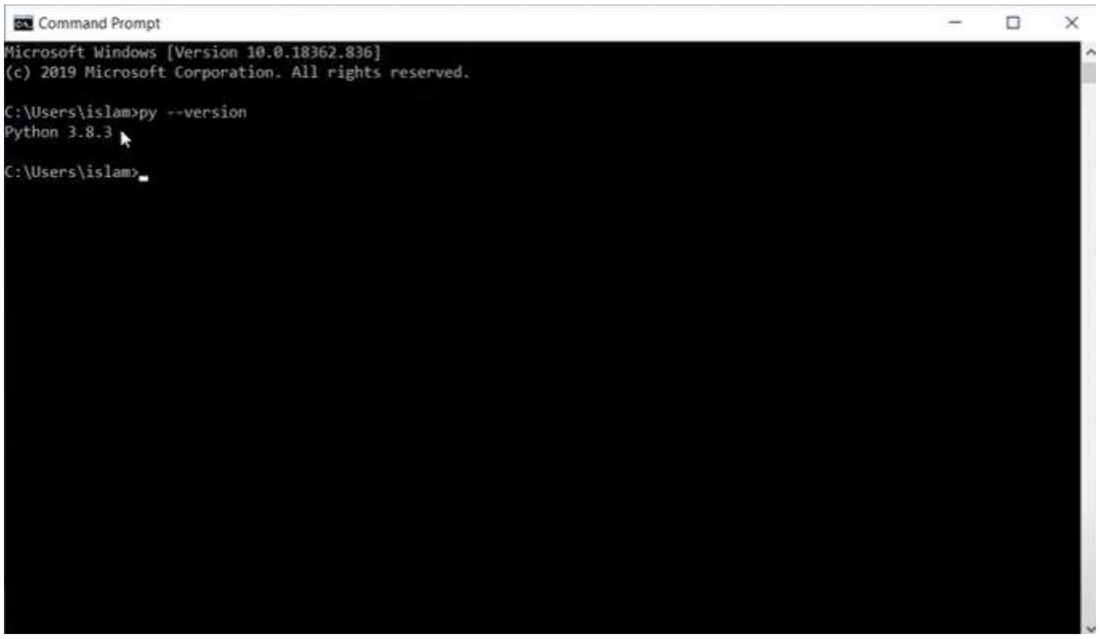
☑ pip
Installs pip, which can download and install other Python packages.

☑ tcl/tk and IDLE
Installs tkinter and the IDLE development environment.

☑ Python test suite
Installs the standard library test suite.

☐ py launcher ☑ for all users (requires elevation)
Upgrades the global 'py' launcher from the previous version.

After choosing your directory, make sure that all of the above options are ticked to be able to run your python files correctly.



```
Microsoft Windows [Version 10.0.18362.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\islam>py --version
Python 3.8.3

C:\Users\islam>
```

Finally, to make sure that python has been installed successfully on your computer, open the terminal window and type py -- version. The version that you have downloaded must be displayed, indicating that Python is installed successfully on your device.

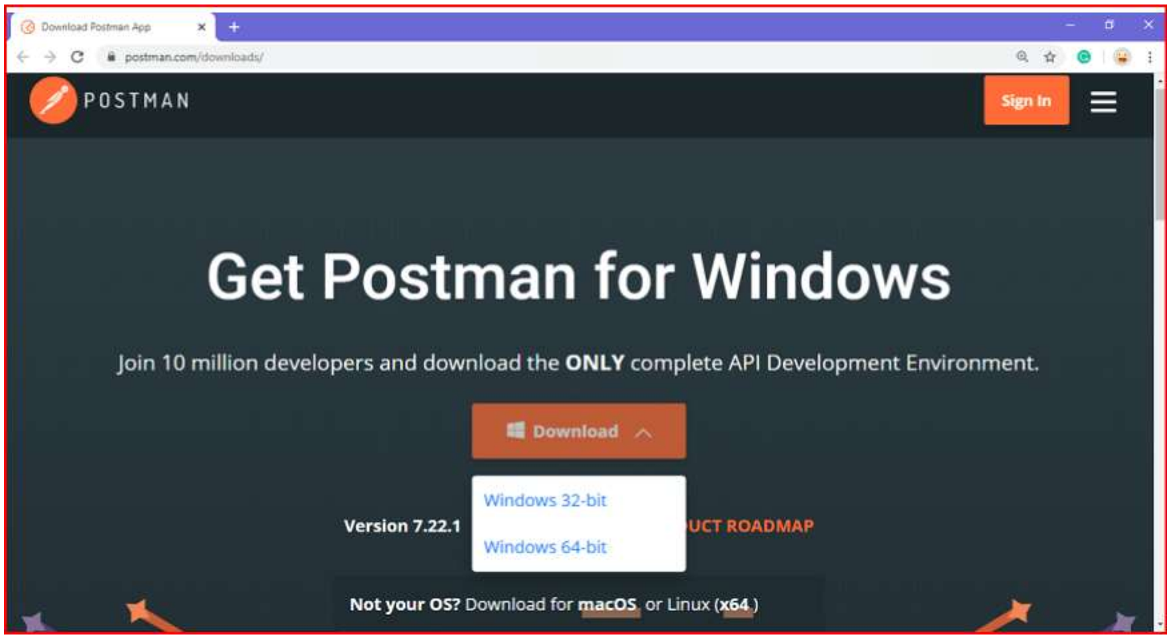
Downloading an IDE (optional):

The program can be executed through the terminal by typing python followed by the filename (Ex: filename.py), however you can download an integrated development environment to ease the process. Below are some of the most popular IDEs used to execute python files:

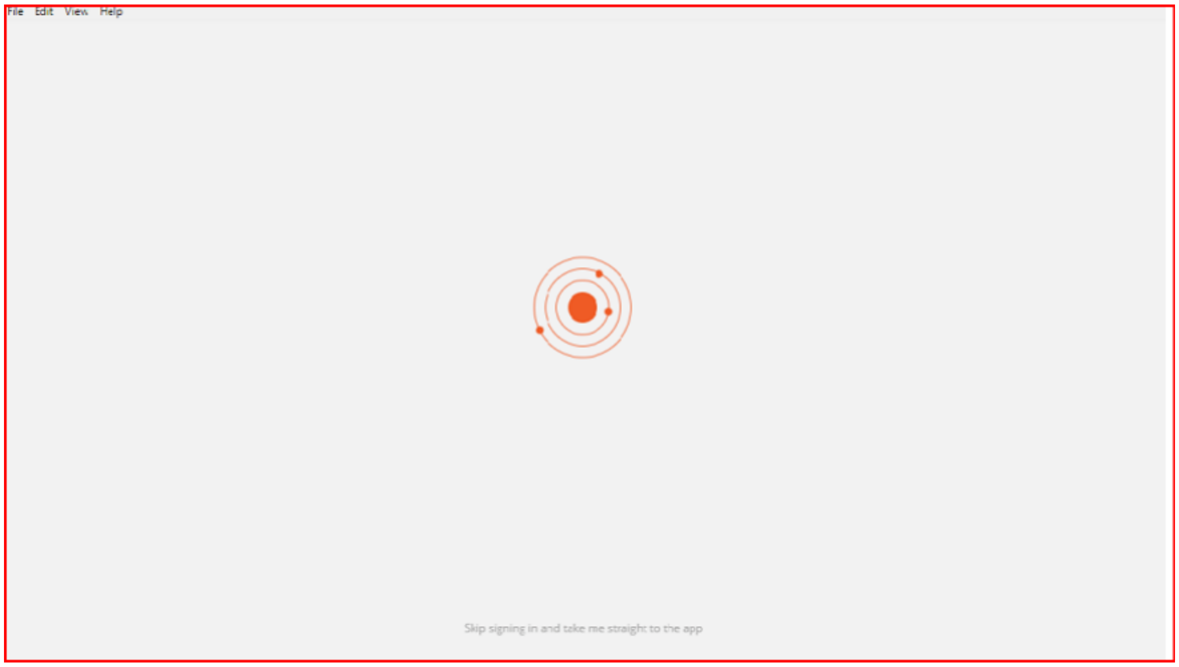
Visual Studio Code, Pycharm, Atom, Spyder…etc.

Downloading Postman API:

Postman is used to interact with our blockchain in the program over the network. To download Postman you go to the link: https://www.postman.com/downloads/ then click download for Mac or Windows or Linux based on your operating system.



Once the installation completes, you will be redirected to a window as shown in the image where you can click on Stop signing in and take me straight to the app (as this app can also be used without logging in) or otherwise you will get a new window to sign up. However, it is better you create an account as this will help you save the work you do within Postman.



Description of the developed Program:

In this system, it is meant to create a crypto-currency application using a decentralized blockchain database system which is consensus-based, including the ability to mine a new block, return the full blockchain or create a new transaction to a block adding to that accepting a list of new nodes and resolving any conflicts to ensure a nod has the correct chain. And this all happens with the use of cryptographic hashing algorithm and digital signature.

User Guide:

The user who uses any of Linux, windows, or OS with the minimum requirements to run any Python IDE, such as Jupyter notebook, Visual studio code, Pycharm, Spyder or IDLE etc., shall be able to install the mentioned IDEs from Chrome browser with the required Python packages to run the program. In addition, because we are dealing with blockchain we need to interact with the network using any API.

After compiling the program, you could connect to the server:

- In our case it is python blockchain.py it tells you running on http://127.0.0.1:5000/ you can press CTRL+C or simply copy and past the link to the API.
- In order to mine a block, you can use GET request: http://localhost:5000/mine
- In order to create a new transaction use POST request: http://localhost:5000/transactions/new
- In order to inspect a full chain, you can request: http://localhost:5000/chain

For connecting two machines you can spin up different nodes previously registered and then considered as neighboring nodes in the network or use different ports on the same machine. In our program we used two different ports which are: http://localhost:5000 and http://localhost:5001.

In addition, there are two more end points the user can implement for the following purposes:

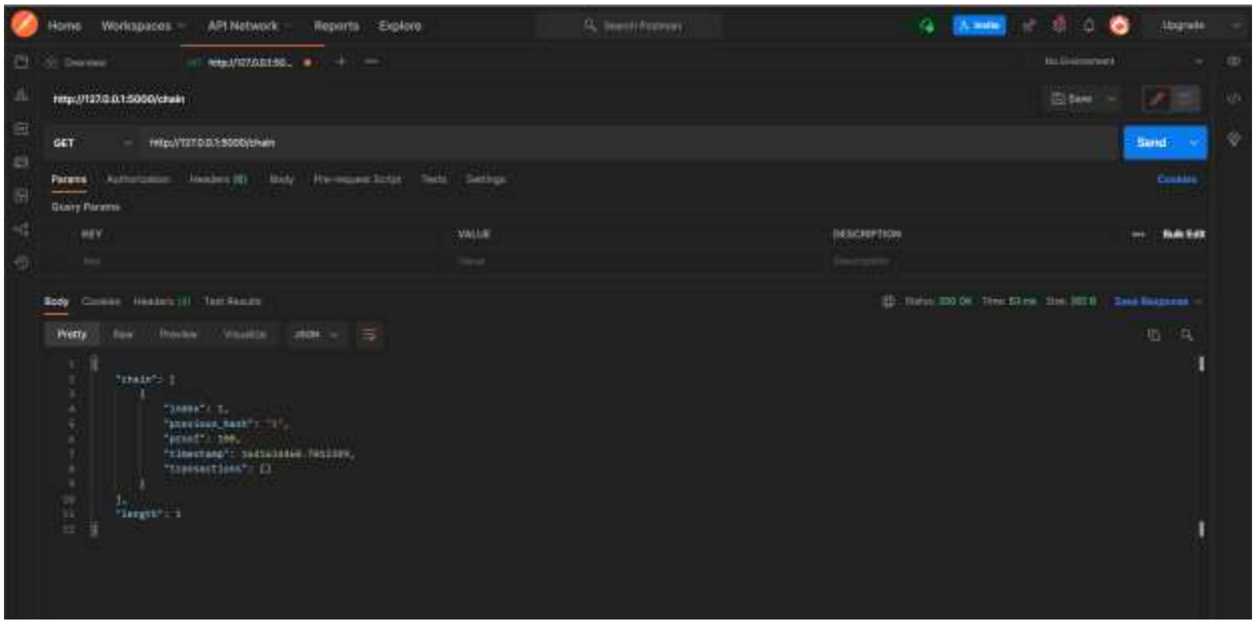
- /nodes/register to accept a list of new nodes in the form of URLs.
- /nodes/resolve to implement our Consensus Algorithm, which resolves

Description of Conducted Tests and their Results:

Both incoming blocks from other nodes and blocks for mining undergo a conformity check before being appended and broadcasted, respectively. This is a two-stage check for block integrity and transaction validity. For checking the block integrity, the signature of the block is checked and the hash of the block is recomputed. Then it is verified whether the block hash satisfies the currently active difficulty level and whether the block is appendable to the current tree. Checking the transaction validity is a more complex process that works as follows:

- Append the block temporarily to the tree and check whether this block changes the longest chain.
- If the longest chain has changed, find the first common parent node for the branch of the previous longest chain and the current longest chain; otherwise proceed with step 5.
- Temporarily revert all transactions (i.e., remove them from the state table) from the previous branch up to the common ancestor (parent) node.
- Process all transactions from the current longest branch starting from the common ancestor node.
- If a single transaction in the newly added block is invalid (e.g., due to double spending or unknown recipients), the whole process is reverted and the new block is discarded.

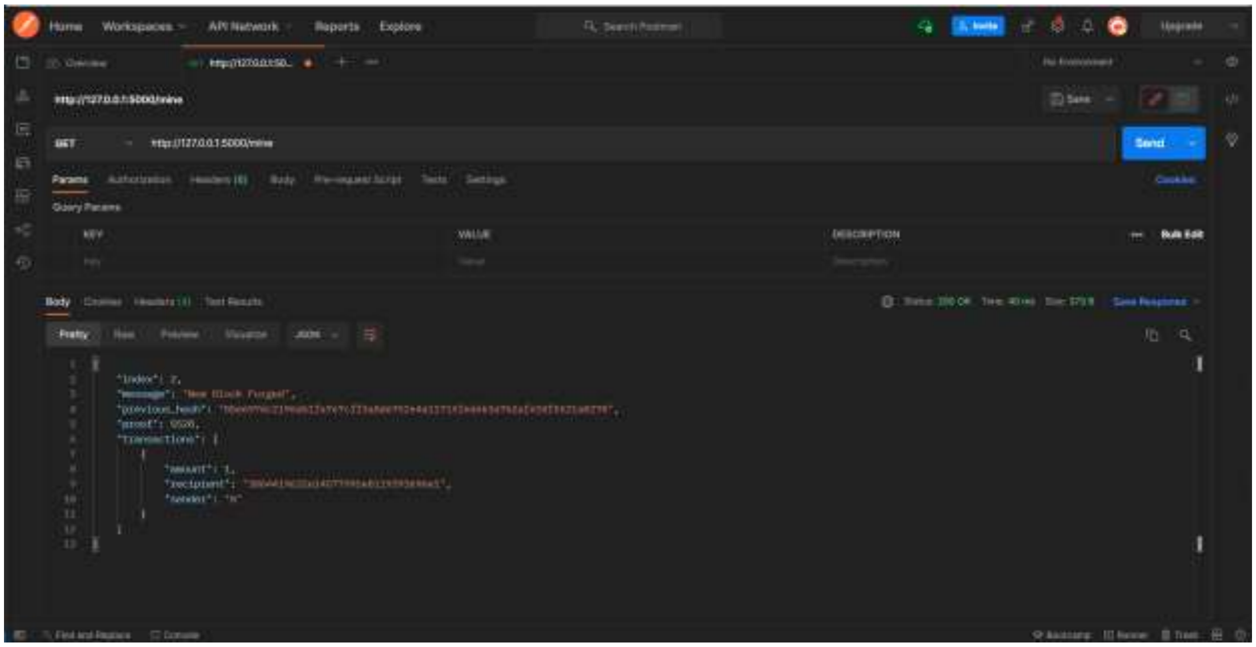
Firstly: we printed the chain from port 5000 as shown.



GET http://127.0.0.1:5000/chain

```
{
    "chain": [
        {
            "index": 1,
            "previous_hash": "1",
            "proof": 100,
            "timestamp": [illegible],
            "transactions": []
        }
    ],
    "length": 1
}
```

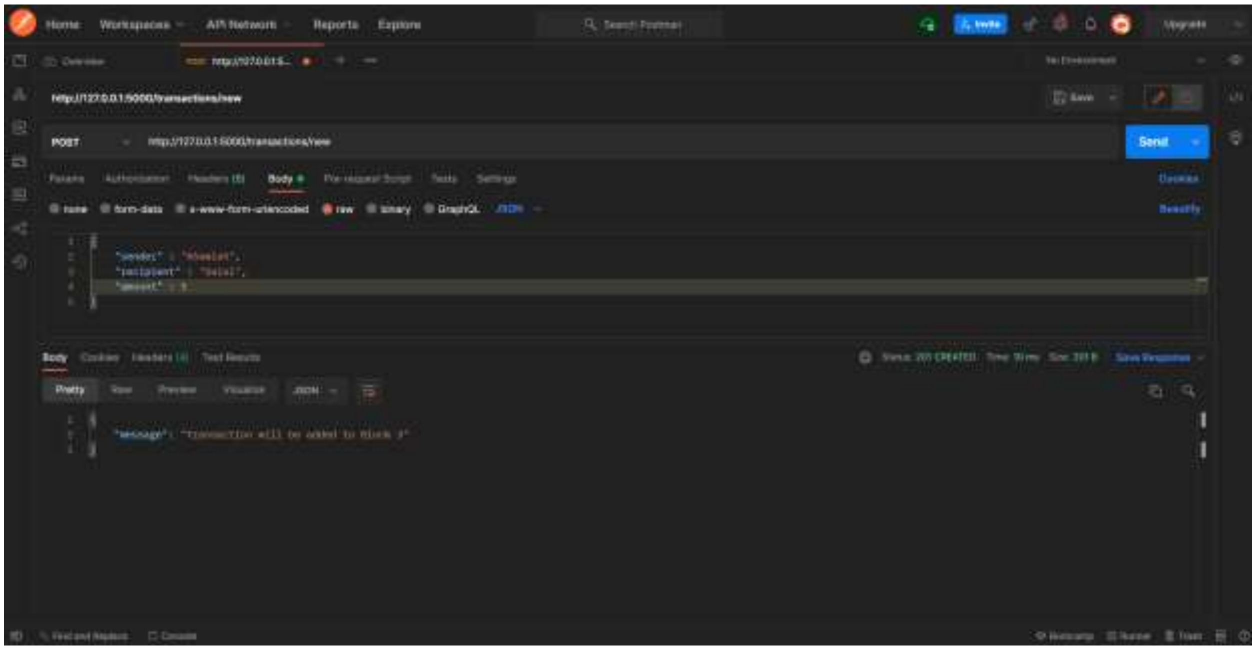
Secondly: mining a new block in port 5000



GET http://127.0.0.1:5000/mine

```
{
    "index": 2,
    "message": "New Block Forged",
    "previous_hash": "[illegible]",
    "proof": [illegible],
    "transactions": [
        {
            "amount": 1,
            "recipient": "[illegible]",
            "sender": "0"
        }
    ]
}
```

Thirdly: we made a new transaction also in port 5000



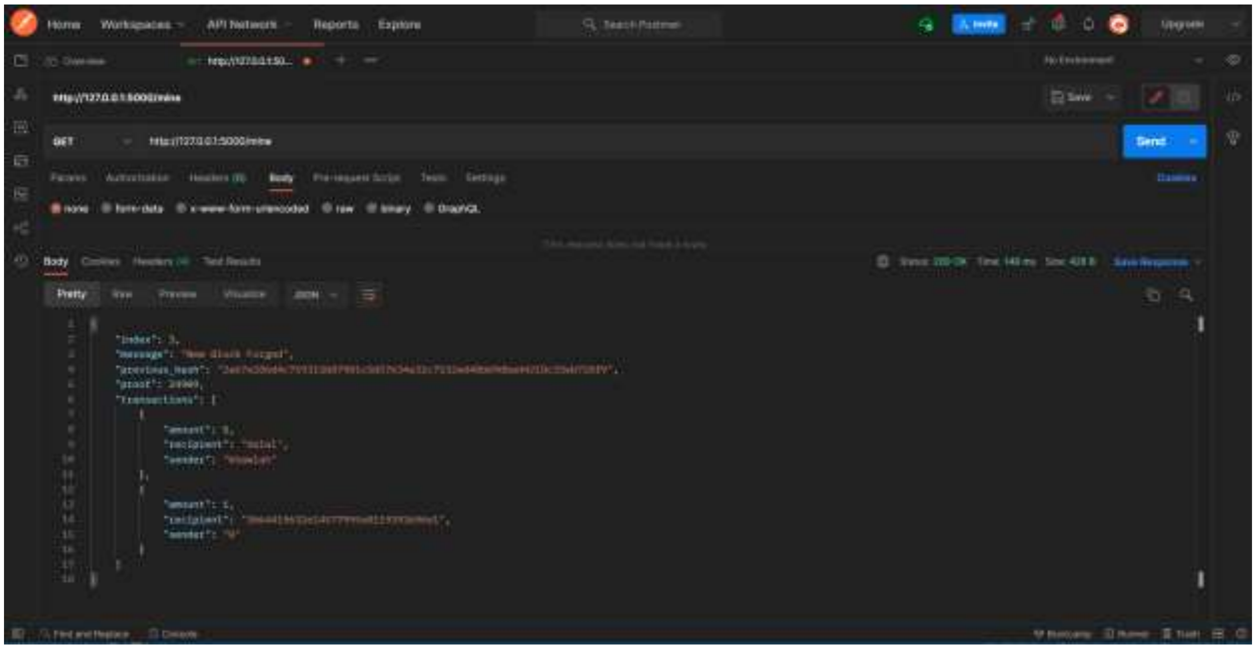
POST http://127.0.0.1:5000/transactions/new

```
{
    "sender" : "Khawlah",
    "recipient" : "Dalal",
    "amount" : 5
}
```

Status: 201 CREATED

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

Fourthly: here we did mining for a new block that transaction was added.

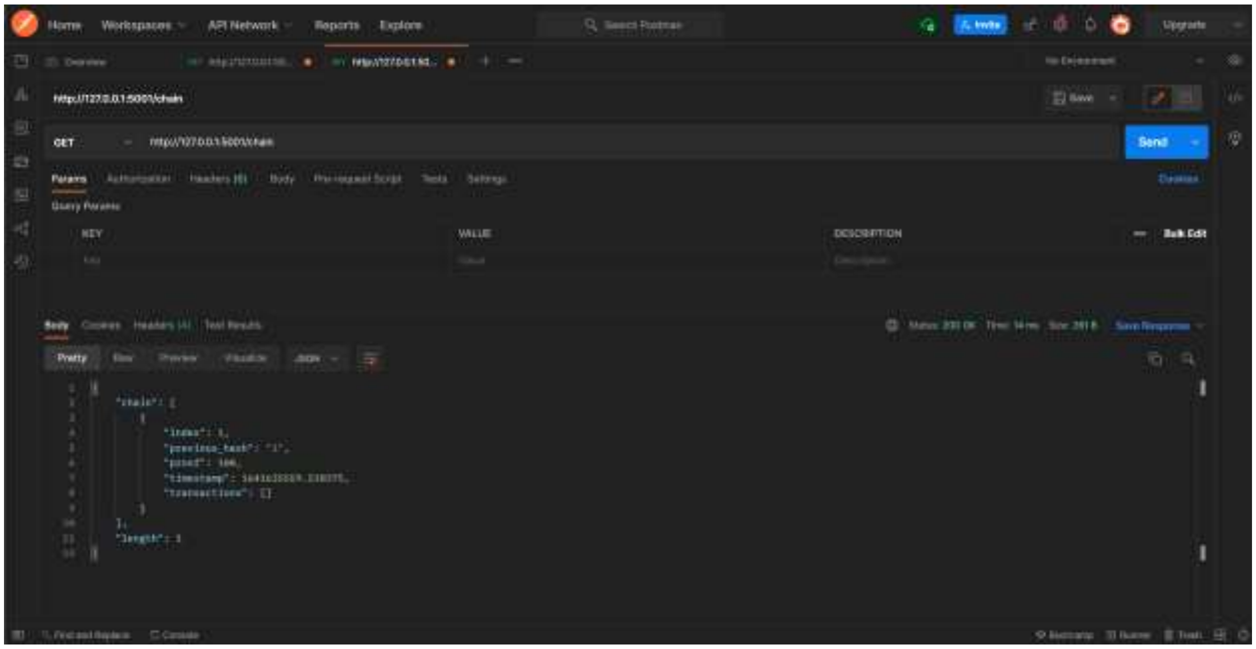


GET http://127.0.0.1:5000/mine

Status: 200 OK

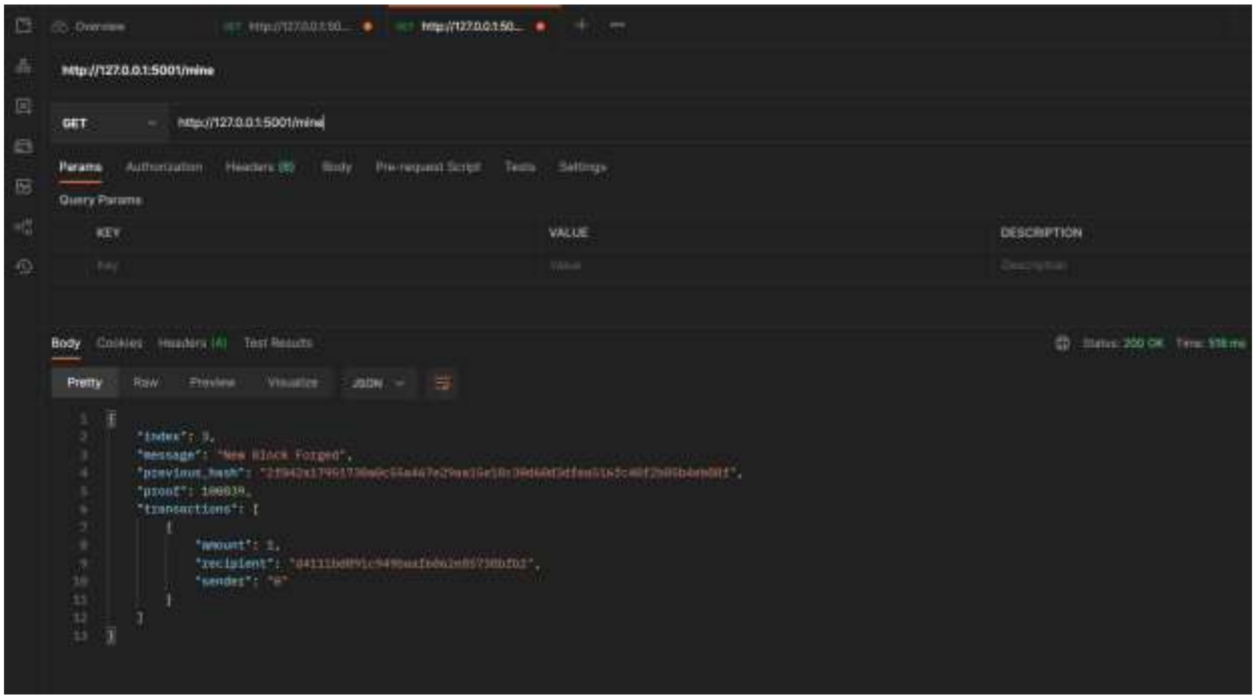
```
{
    "index": 3,
    "message": "New Block Forged",
    "previous_hash": "[illegible]",
    "proof": 29909,
    "transactions": [
        {
            "amount": 5,
            "recipient": "Dalal",
            "sender": "Khawlah"
        },
        {
            "amount": 1,
            "recipient": "[illegible]",
            "sender": "0"
        }
    ]
}
```

Then we tested printing the chain in port 5001.



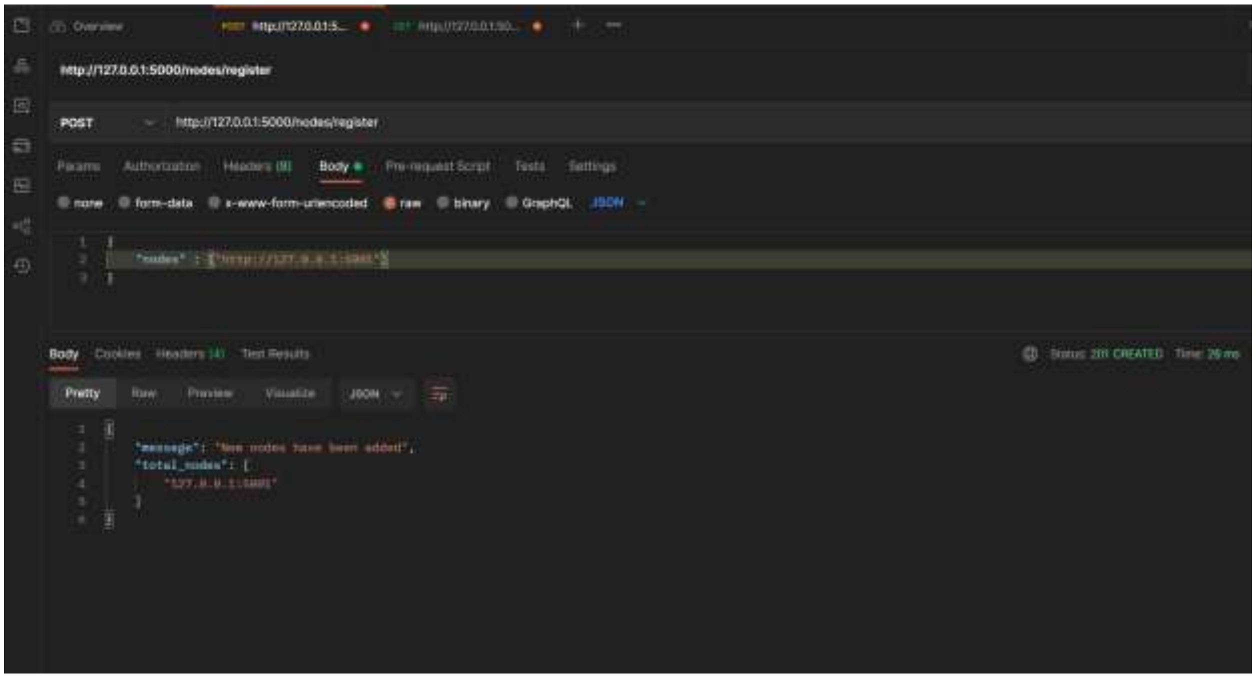
```
{
    "chain": [
        {
            "index": 1,
            "previous_hash": "1",
            "proof": 100,
            "timestamp": [illegible],
            "transactions": []
        }
    ],
    "length": 1
}
```

After that we mined two more blocks in the new port (5001) in a way that there are three blocks in each port.



```
{
    "index": 3,
    "message": "New Block Forged",
    "previous_hash": "[illegible]",
    "proof": 100838,
    "transactions": [
        {
            "amount": 1,
            "recipient": "[illegible]",
            "sender": "0"
        }
    ]
}
```

Here we registered node 5001 in 5000:



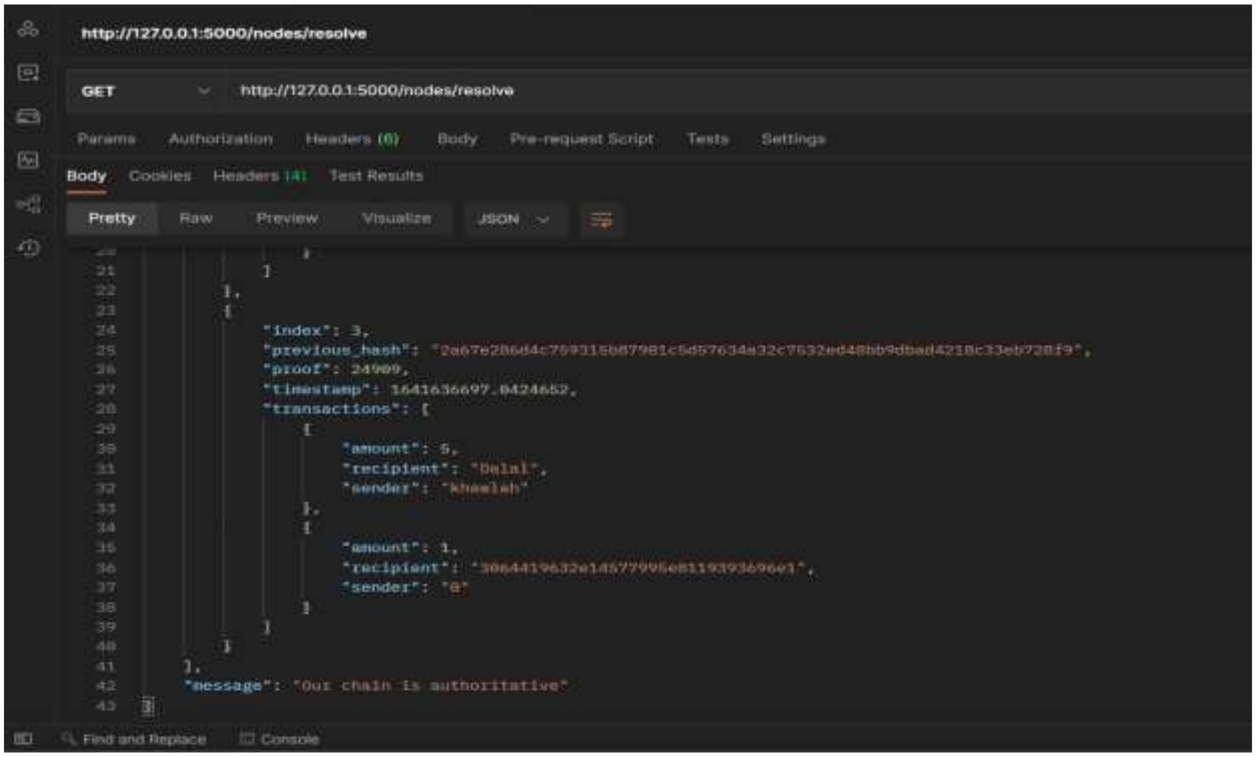
POST http://127.0.0.1:5000/nodes/register

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

Status: 201 CREATED

```
{
    "message": "New nodes have been added",
    "total_nodes": [
        "127.0.0.1:5001"
    ]
}
```

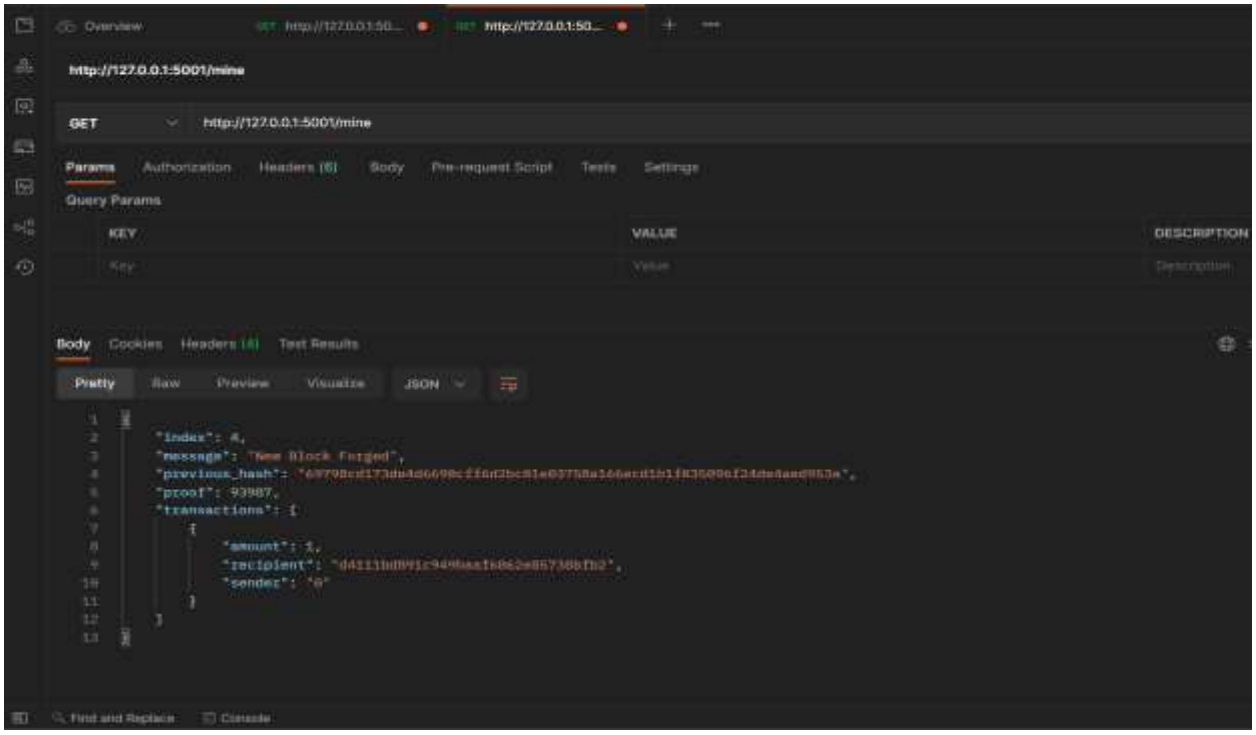
Then we tested calling the nodes/resolve endpoint to validate both ports. Since both ports have same no of blocks in their chain, the printed message is "the chain is authoritative":



GET http://127.0.0.1:5000/nodes/resolve

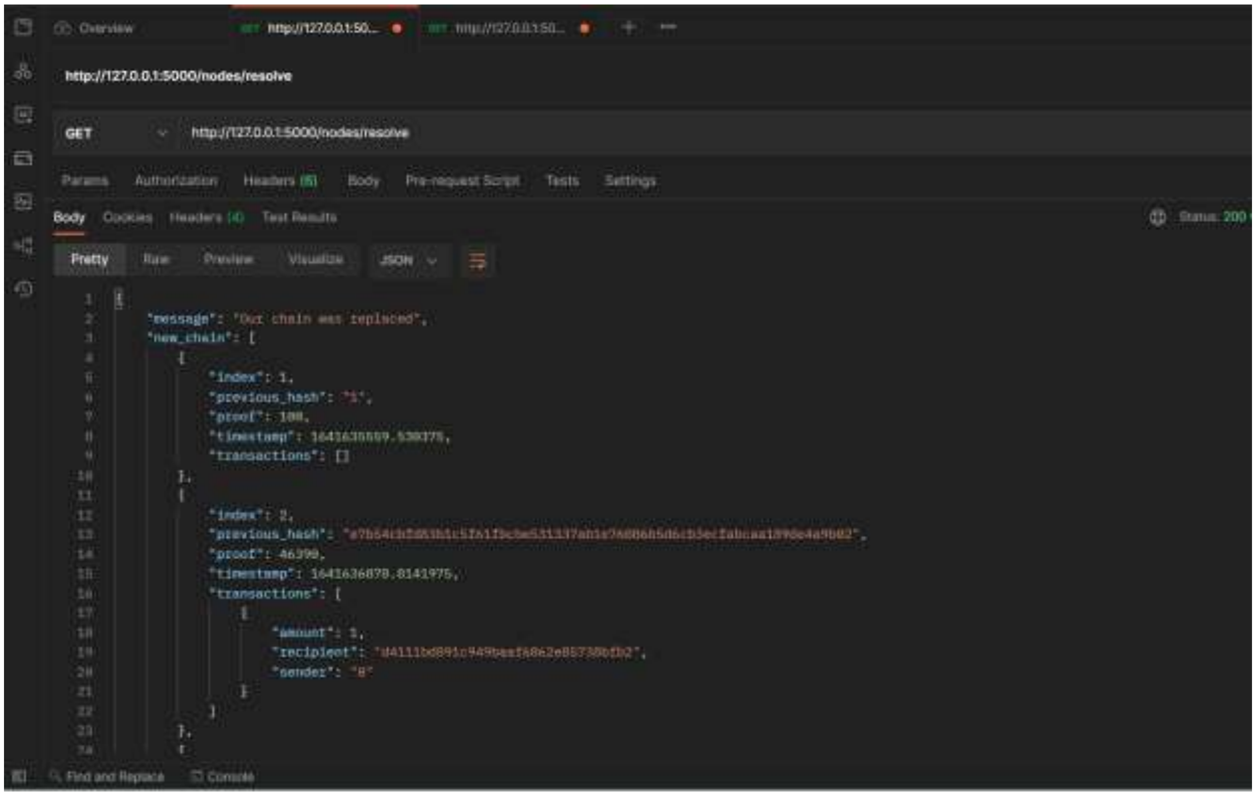
```
                ]
            },
            {
                "index": 3,
                "previous_hash": "2a67e286d4c759315b87981c5d57634a32c7632ed48bb9dbad4218c33eb728f9",
                "proof": 24909,
                "timestamp": 1641636697.0424652,
                "transactions": [
                    {
                        "amount": 5,
                        "recipient": "Dalal",
                        "sender": "Khawlah"
                    },
                    {
                        "amount": 1,
                        "recipient": "30644196632e14577995e81193936 96e1",
                        "sender": "0"
                    }
                ]
            }
        ],
        "message": "Our chain is authoritative"
}
```

Here we mined a new block in port 5001 which makes it have one more block to check the validity of replacing the longest chain in each of the ports.



```
{
    "index": 4,
    "message": "New Block Forged",
    "previous_hash": "69798cd173de4d6698cff6d2bc81e03758a166acd1b1f835096f34de4aed953e",
    "proof": 93987,
    "transactions": [
        {
            "amount": 1,
            "recipient": "d4111bd891c949baaf6862e857380fb2",
            "sender": "0"
        }
    ]
}
```

Now we tested calling the resolve endpoint again in which our chain has been replaced by the longest one:



```
{
    "message": "Our chain was replaced",
    "new_chain": [
        {
            "index": 1,
            "previous_hash": "1",
            "proof": 100,
            "timestamp": 1641635559.530375,
            "transactions": []
        },
        {
            "index": 2,
            "previous_hash": "e7b54cb5d63b1c5fa1fbcbe531337ab1e7a86b65d6cb3ecfabcaa1896e4a9b82",
            "proof": 46399,
            "timestamp": 1641636878.8141975,
            "transactions": [
                {
                    "amount": 1,
                    "recipient": "d4111bd891c949baaf6862e857380fb2",
                    "sender": "0"
                }
            ]
        },
        {
```

By this we have implemented all tests applicable to ensure the system we have created in working properly.

Conclusion:

We presented our secure cryptocurrency implementation for fulfilling cryptographic security interacting with networks. We have described the way we coded, run and compiled the program as we elaborated on the connection of two networks using ports. In our project, we tried implementing what we learned throughout the semester regarding security of systems as we used Merkle tree for hashing purposes in blockchain which is used to efficiently verify the integrity of data of transactions on the network. Hashing functions maintain the structure of blockchain data and encode user's account addresses, which is an integral part of the process of encrypting transactions that occur between accounts, and makes block mining possible.

References:

- *Build software better, together*. (n.d.). GitHub. https://github.com/topics/cryptocurrency
- *Freqtrade/freqtrade: Free, open source crypto trading bot*. (n.d.). GitHub. https://github.com/freqtrade/freqtrade
- *Helmi/awesome-crypto: An awesome list about everything crypto currency*. (n.d.). GitHub. https://github.com/Helmi/awesome-crypto
- *Implementation of cryptocurrency using smart contract*. (n.d.). C# Corner - Community of Software and Data Developers. https://www.c-sharpcorner.com/article/implementation-of-cryptocurrency-using-smart-contract/
- Selmanovic, D. (2015, August 7). *Cryptocurrency for dummies: Bitcoin and beyond*. Toptal Engineering Blog. https://www.toptal.com/bitcoin/cryptocurrency-for-dummies-bitcoin-and-beyond