**Pre-installation Requirement**

Postman HTTP Client: <https://www.getpostman.com/>

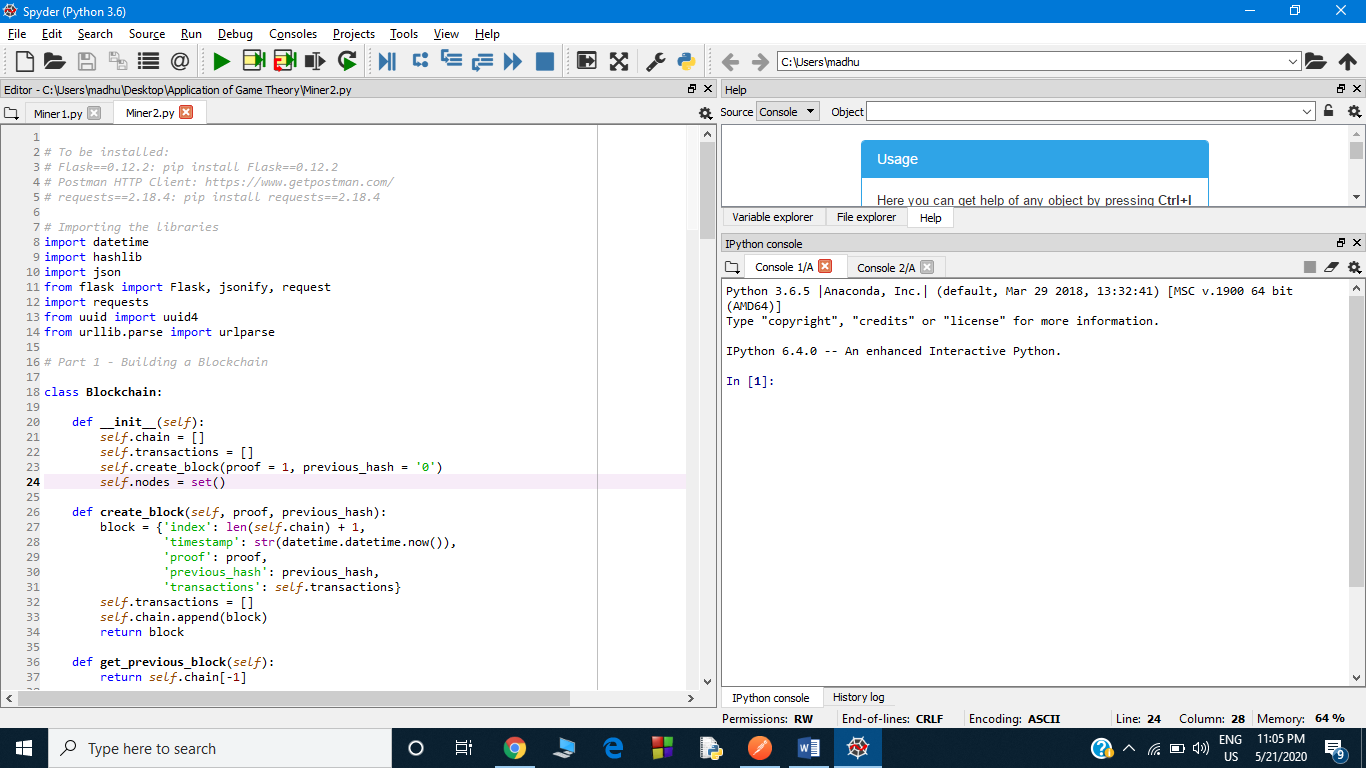
To be installed on conda command prompt:

Flask==0.12.2 using command: pip install Flask==0.12.2

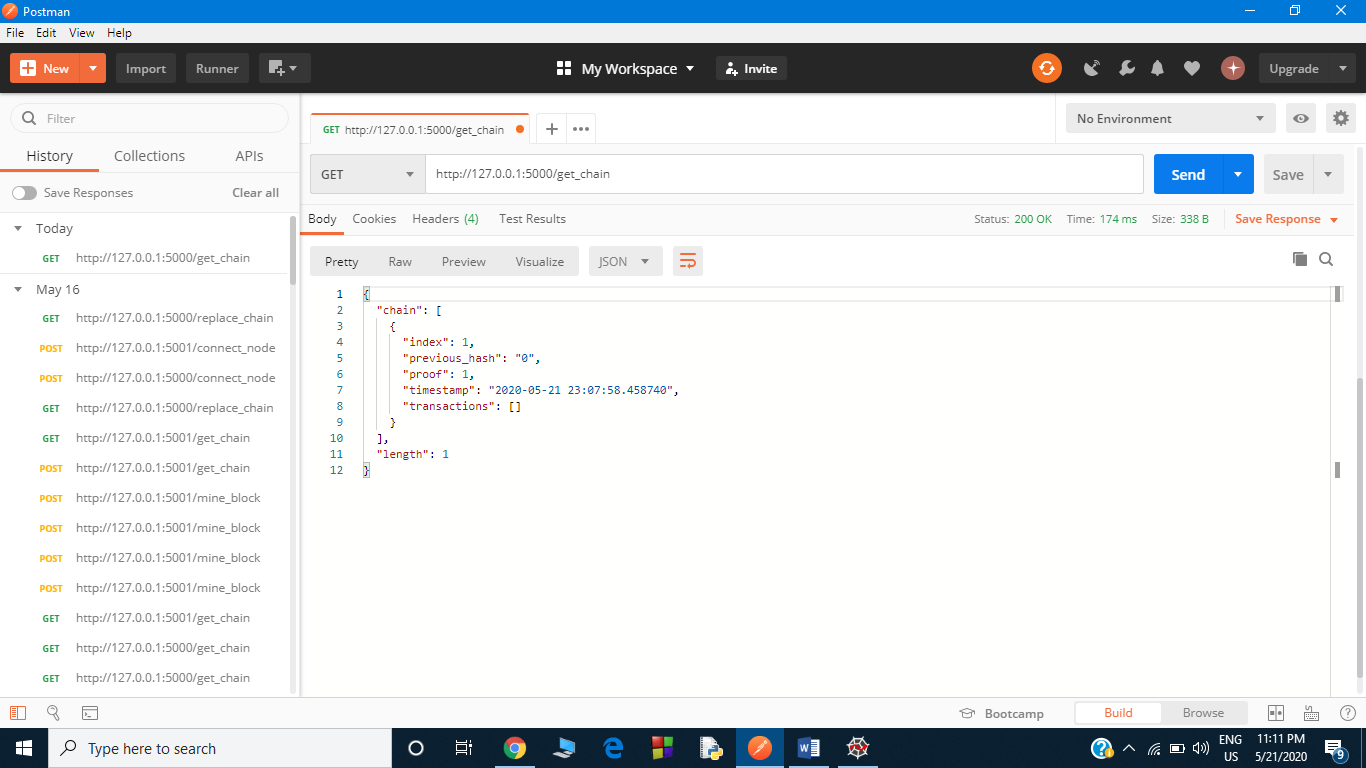
requests==2.18.4 using command: pip install requests==2.18.4

**Steps for implementation’s execution**

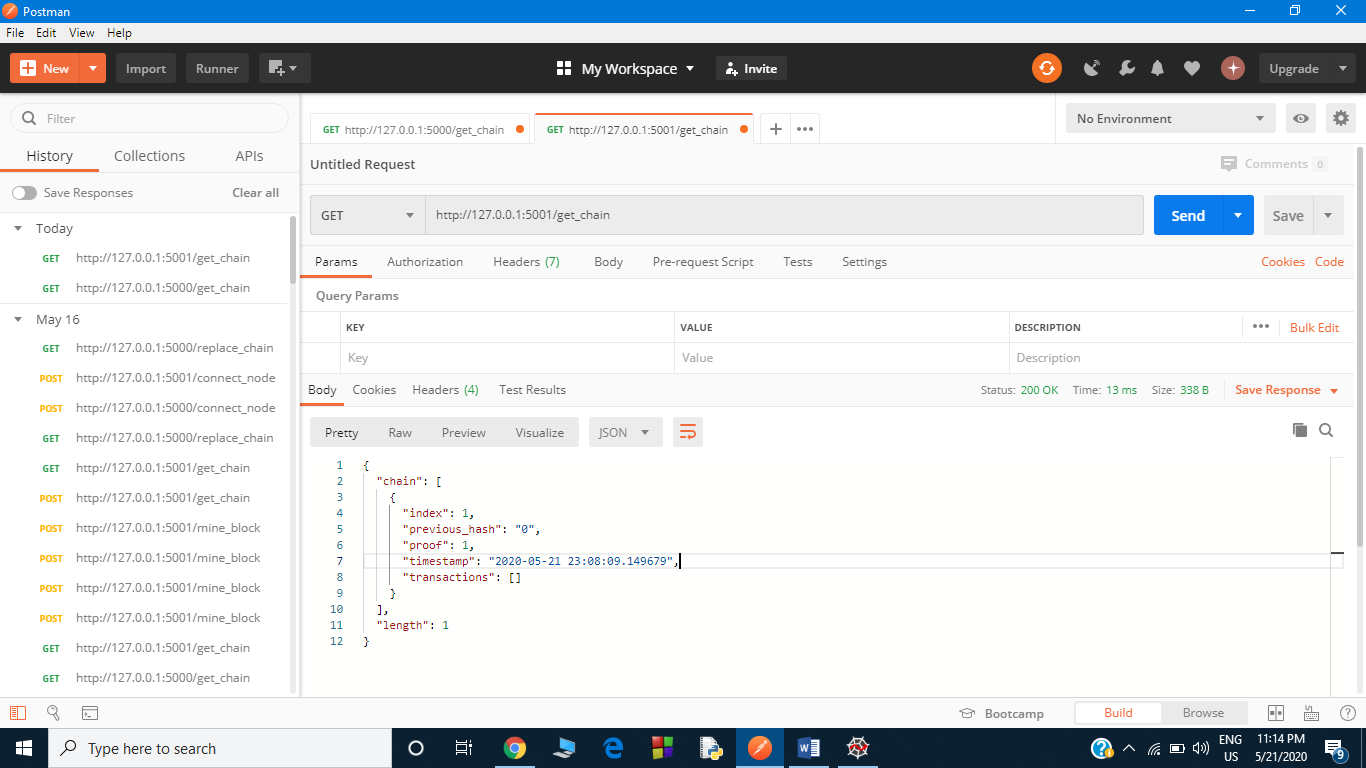
* Open codes, Miner1.py and Miner2.py on Spyder(Python 3.6).
* Create two consoles for our codes execution i.e., Console 1/A and Console 2/A (Command **Ctrl+T** could be used to open 2nd console). After doing all these it should look like as shown below:



* Now run Miner1.py on Console 1/A and Miner2.py on Console 2/A.
* Open Postman.
* Now on postman run the following GET command to see whether everything is fine in our code : http://127.0.0.1:5000/get\_chain and it should look like this:



* Now open another tab on postman to run our miner2 i.e., port 5001 and it could be done by opening another tab and running GET command : http://127.0.0.1:5001/get\_chain and it should now look like:



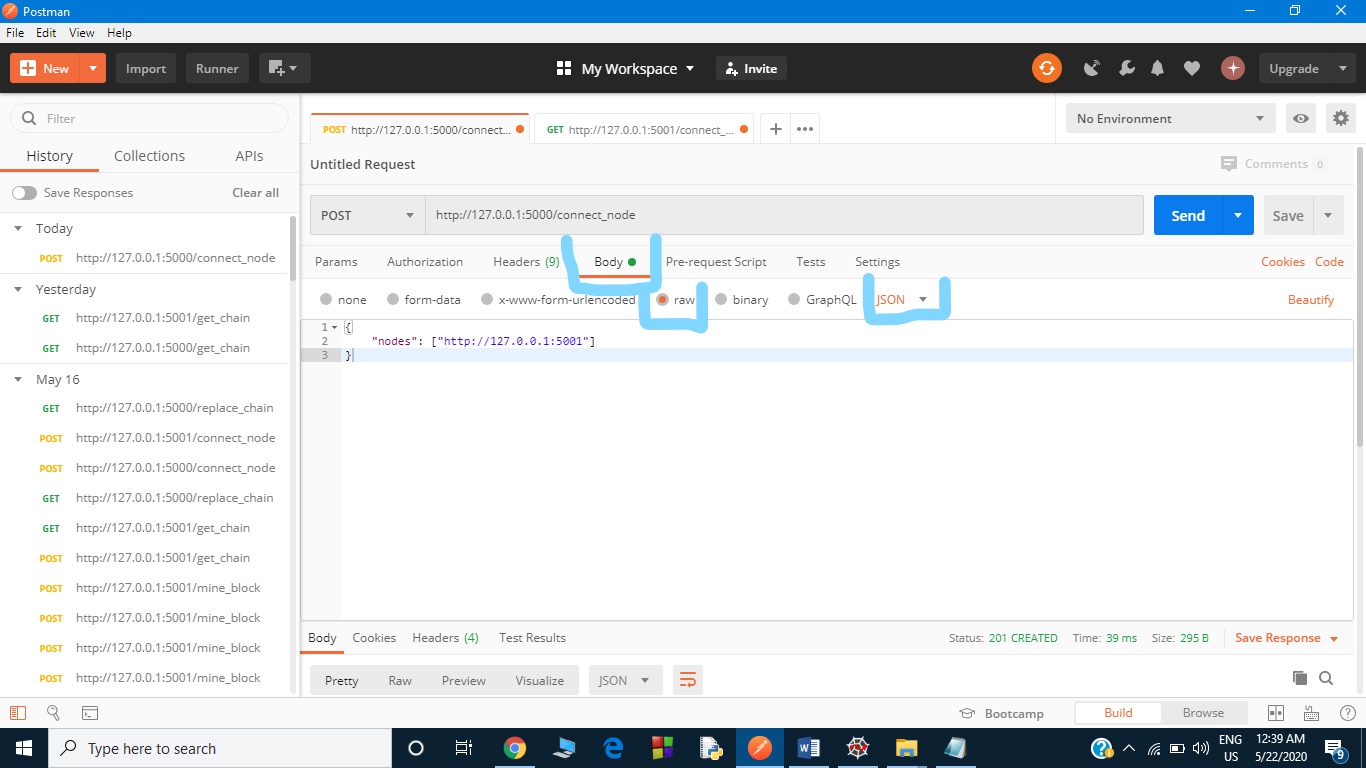
* Now we have miner1 running on port 5000 and miner2 on port 5001.

**Note: to run anything through miner1 our link address must be of the form http://127.0.0.1:5000/FUNCTION-NAME and must be executed on tab corresponding to miner1 on postman i.e., 1st tab.**

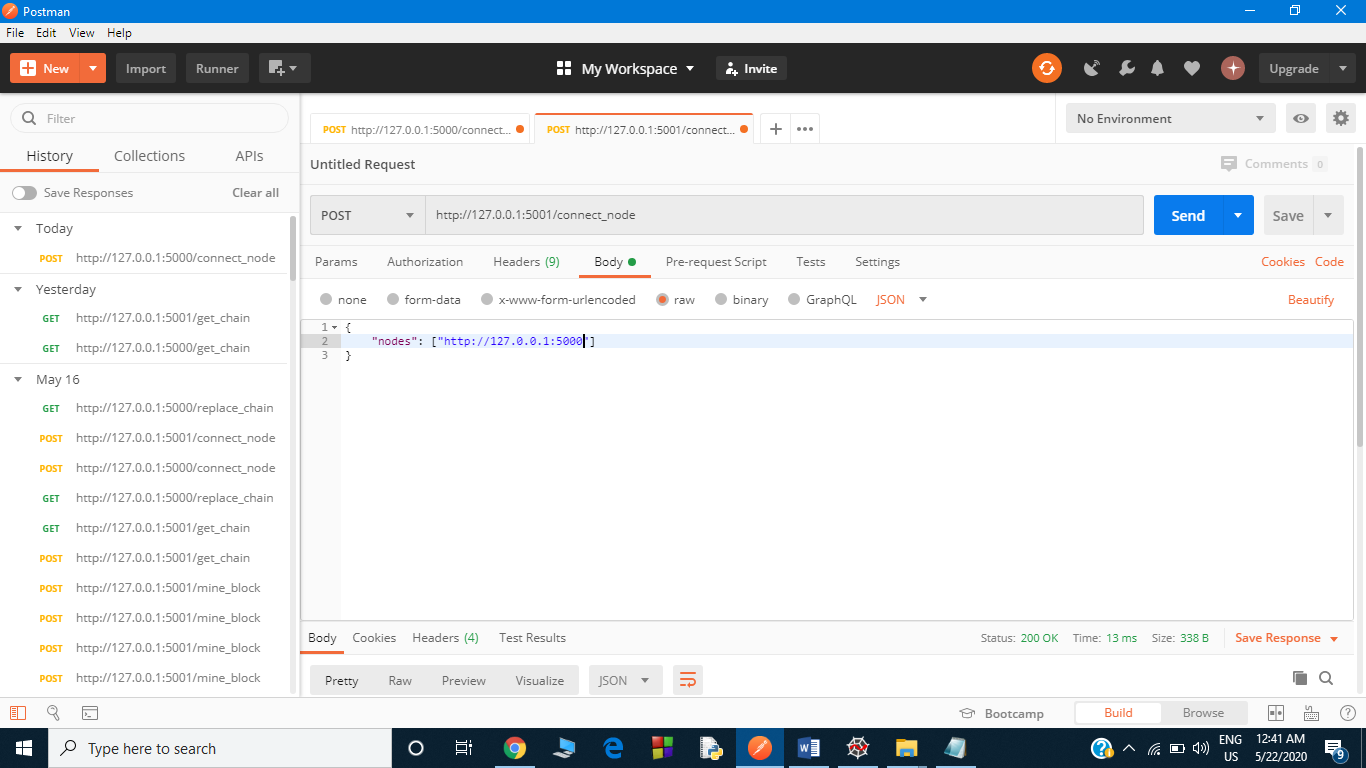
Now follow steps mentioned below and interact with blockchain we have created:

**Connect Nodes:**

Open tab corresponding port 5000 in postman and run following POST method by adding JSON file in the body of the request, we have **nodes.json** created for this just copy and paste it in the body as shown below (**Select the following options as highlighted below**):

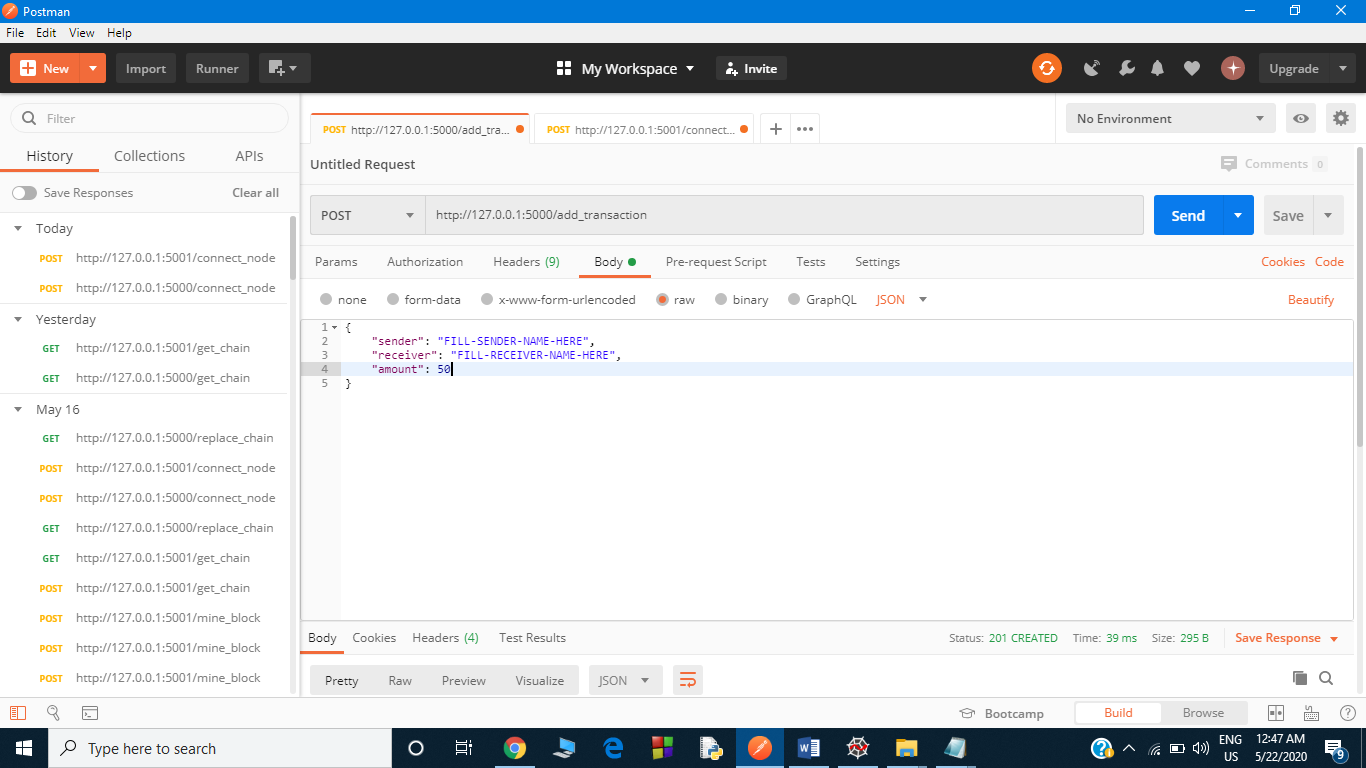


After this port 5000 now knows about port 5001, now we have to repeat same procedure for port 5001 and in JSON file nodes.json we have to now update address from 5001 to 5000 as we have to connect 5000 with 5001 follow the following figure:



**Add transaction:**

Any number of transactions could be added and any miner could add that. It is POST method so we have to add JSON file at the body and update sender name, receiver’s name and amount to be transferred in the JSON file and add transaction (use **transaction.json )** for example miner1 running on port 5000 could add transaction as shown below:



Any numbers of transactions could be added and by any miner.

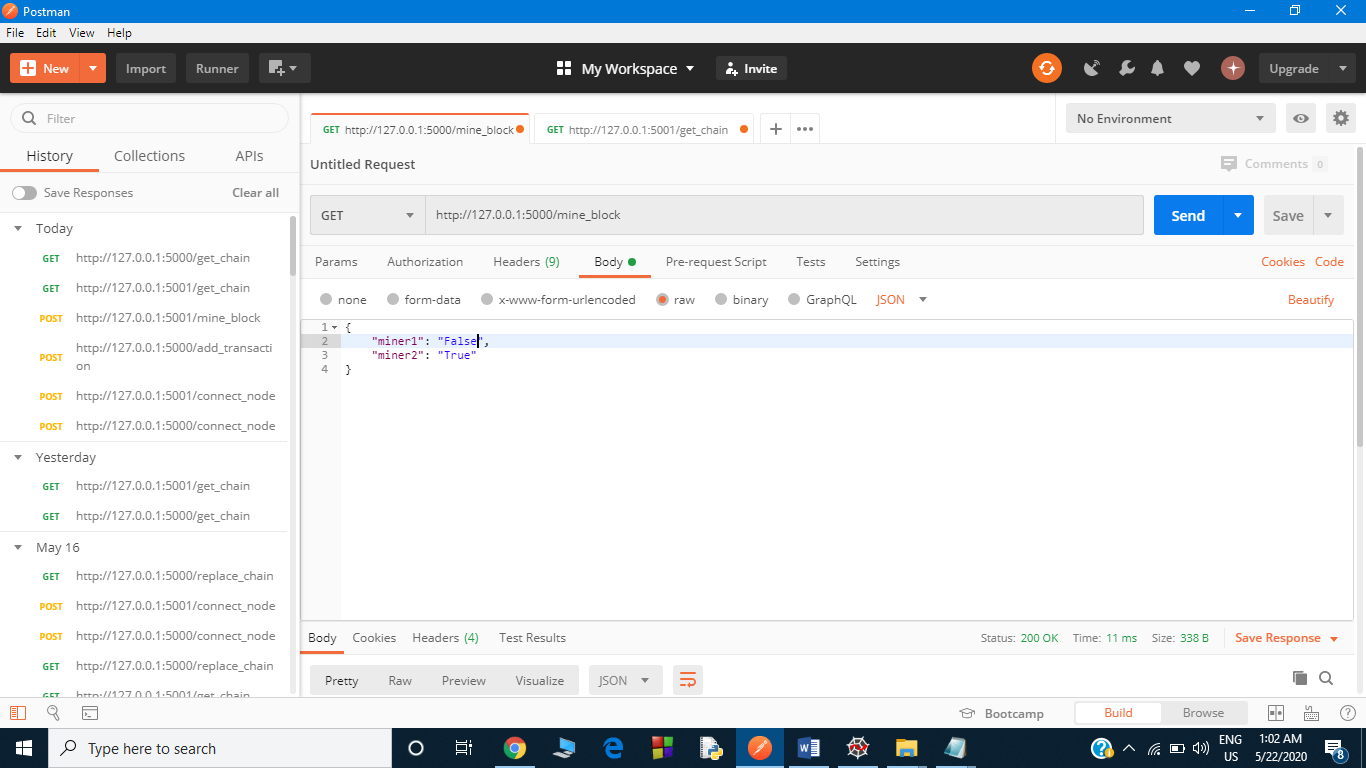
**Mine block**

It is also a POST request and required JSON file in its body (**use Miner’s\_loyalty.json** ).

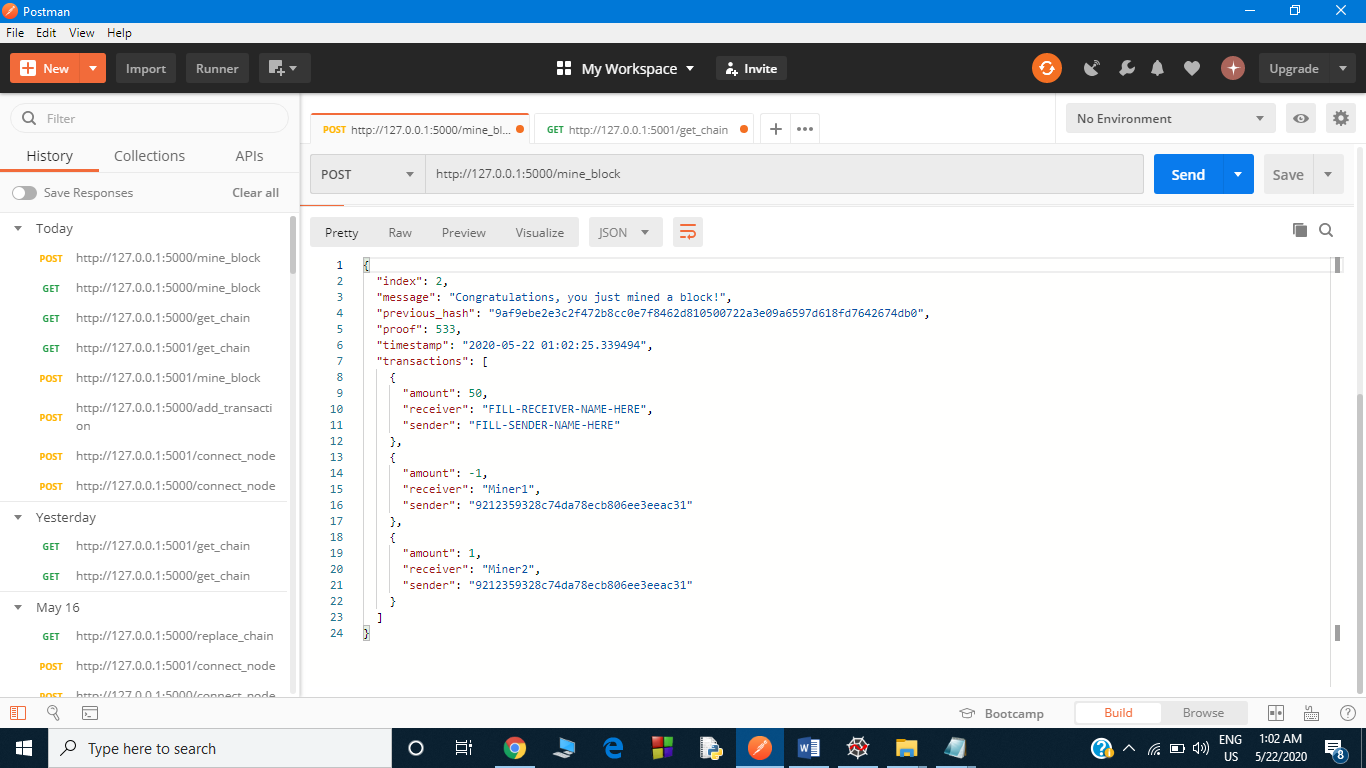
In this JSON file miner1 could be true or false and miner2 could also be true or false and reward is distributed according to the scenario. One may try changing JSON files value and see how reward distributes differently.

An example of miner1 mining the block with miner1’s loyalty as False and miner2’s loyalty as True.

**Note: the input is case sensitive so use True/False**



Input



Output

As at this moment miner1 has updated chain but miner2 doesn’t since we have not yet excecuted the function running consensus.

We can see the that miner1’s chain and miner2’s chain is different by running

http://127.0.0.1:5000/get\_chain on miner1’s tab

http://127.0.0.1:5001/get\_chain on miner2’s tab

As miner2 has different node so we have to apply our consensus on miner2 and this could be done by running http://127.0.0.1:5001/replace\_chain on miner2’s tab.

Now we can add transactions, mine blocks, see chain present using get\_chain, update the maximum length chain by using replace\_chain method.

**Note: add\_transaction , mine\_block and connect\_node are POST request so we have to add corresponding JSON file to the body of request, rest all are GET requests.**