Simple Blockchain using Proof of Work by **Anirudh Bhattacharya**

Abstract:
Blockchain is a decentralized database that contains copies of all conducted transactions and other digital events that were shared among participants. Every transaction is confirmed by the vast majority of system users. It includes every single transaction record.
In my project, I have created a chain with adequate security and four nodes and can be used on a single computer as well as on multiple ones.

Introduction:
Blockchain is a newly emerging technology, which although has been popularized by cryptocurrencies such as Bitcoin and Etherium, it may be hard to understand for some. Thus, the goal of this project is to make bitcoin easy to implement and understand.

This app does the following functions:

1. Create chain
2. Validate blocks
3. Implement Byzantine fault tolerance.
4. Check hashes
5. Use SHA-256

Requirements:
Hardware:

1. Dual core processor
2. 2GB RAM
3. At least 45kB of storage space

Software:

1. Windows 7+ / Ubuntu
2. Flask
3. Postman

Proposed System:
This system will make blockchain technology easy to understand and implement. One can build more complex applications on top of this system: such as a blood bank system, a banking system and even a Non Fungible Token system.

Economic Feasibility:
This system will only require the cost of web hosting to be fulfilled.

Technical Feasibility:
This system has a simple structure and has the following requirements:
Flask==2.2.2

Operational Feasibility:
This system has performed with adequate speed and with minimal bugs.

Software Specifications:

Flask:

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy.

There are many modules or frameworks which allow building your webpage using python like a bottle, Django, Flask, etc. But the real popular ones are Flask and Django. Django is easy to use as compared to Flask but Flask provides you with the versatility to program with.

To understand what Flask is you have to understand a few general terms.

- WSGI Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.
- Werkzeug It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.
- jinja2 jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.
- Flask is a web application framework written in Python. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Postman:

Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

You can send requests in Postman to connect to APIs you are working with. Your requests can retrieve, add, delete, and update data. Whether you are building or testing your own API, or integrating with a third-party API, you can send your requests in Postman. Your requests can send parameters, authorization details, and any body data you require.

For example, if you're building a client application (such as a mobile or web app) for a store, you might send one request to retrieve the list of available products, another request to create a new order (including the selected product details), and a different request to log a customer in to their account.

When you send a request, Postman displays the response received from the API server in a way that lets you examine, visualize, and if necessary troubleshoot it.

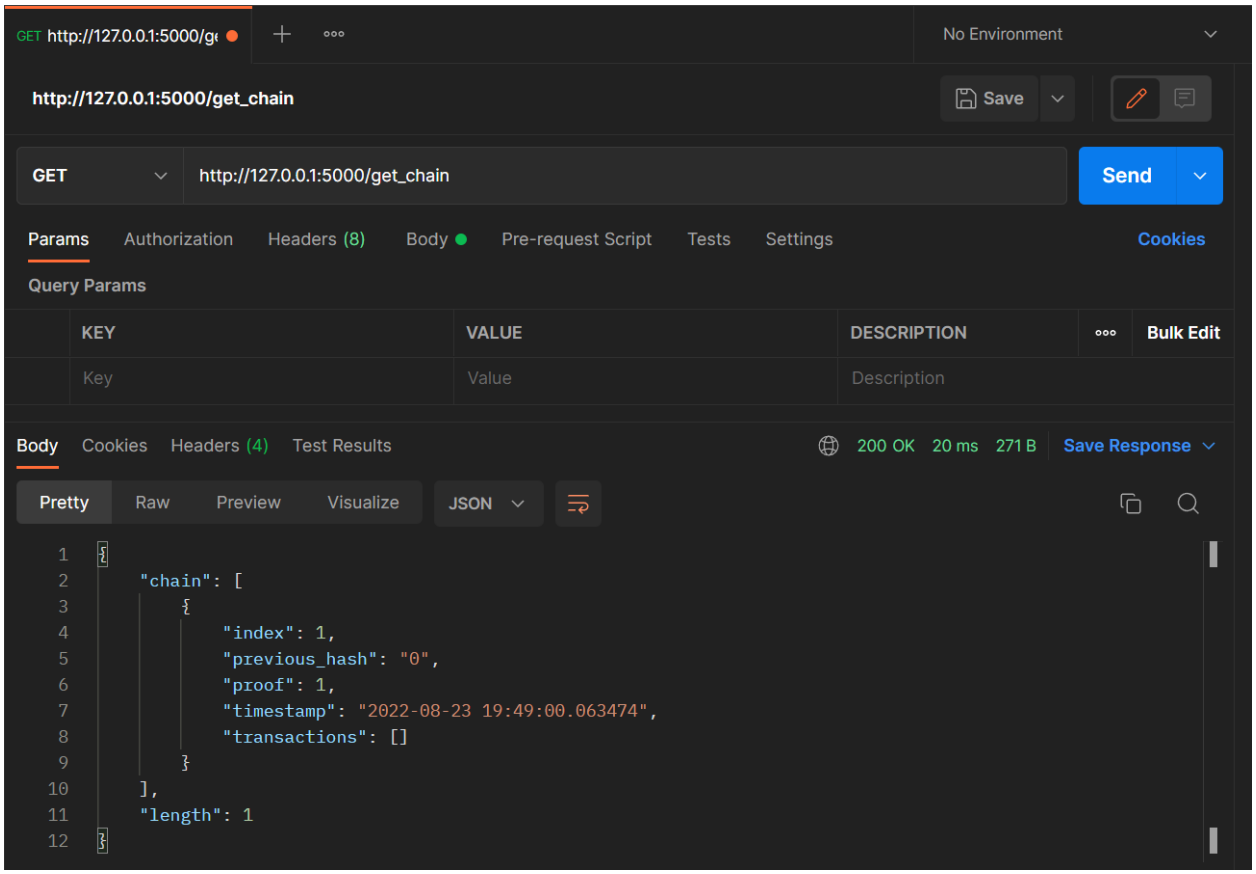
Implementation:

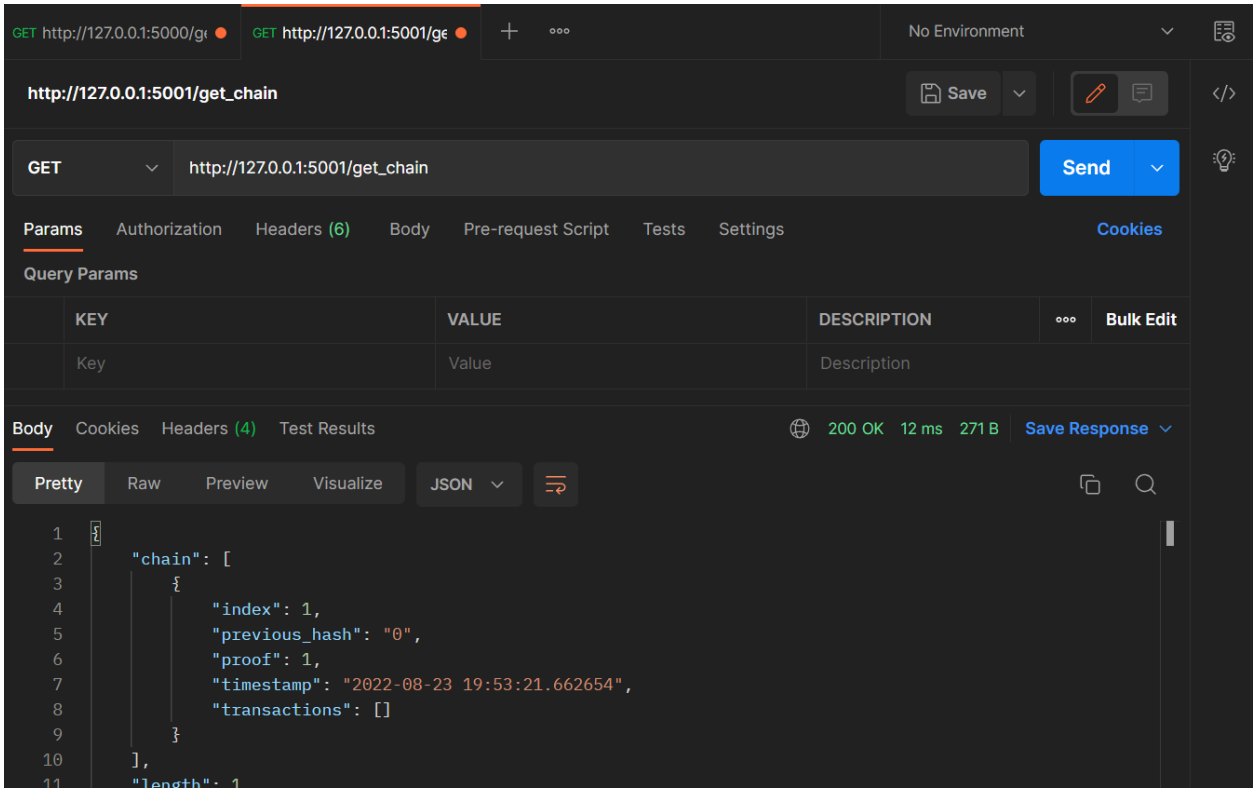
How to run:

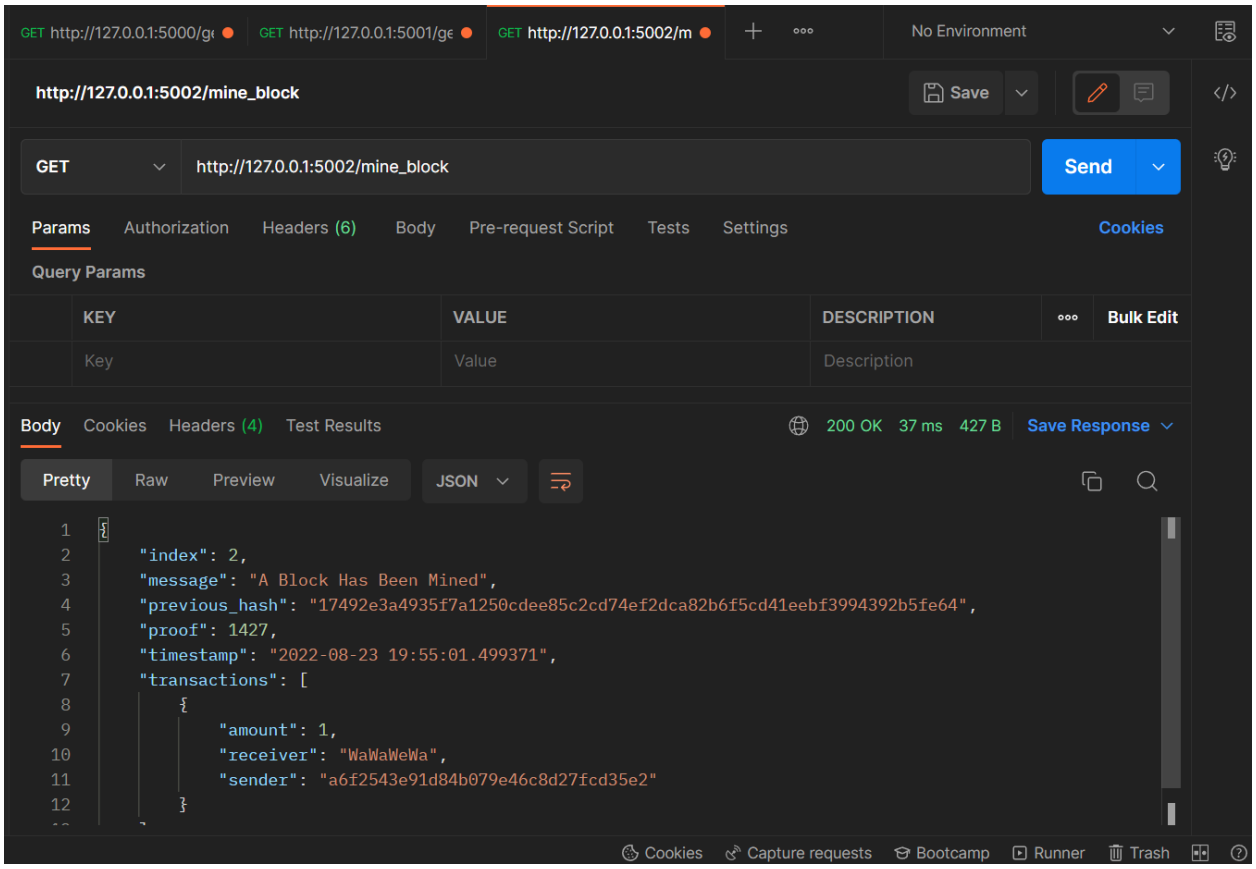
1. Run all of the Python files in the folder.
2. Launch Postman.
3. Select GET and paste http://127.0.0.1:5000/ in the input box.
4. Similarly, paste http://127.0.0.1:5001/, http://127.0.0.1:5002/, http://127.0.0.1:5003/ in the input box.
5. Use the get_chain or the mine_block methods to either view the blockchain or to mine the blocks.
6. The chain will be updated as every user mines a block.

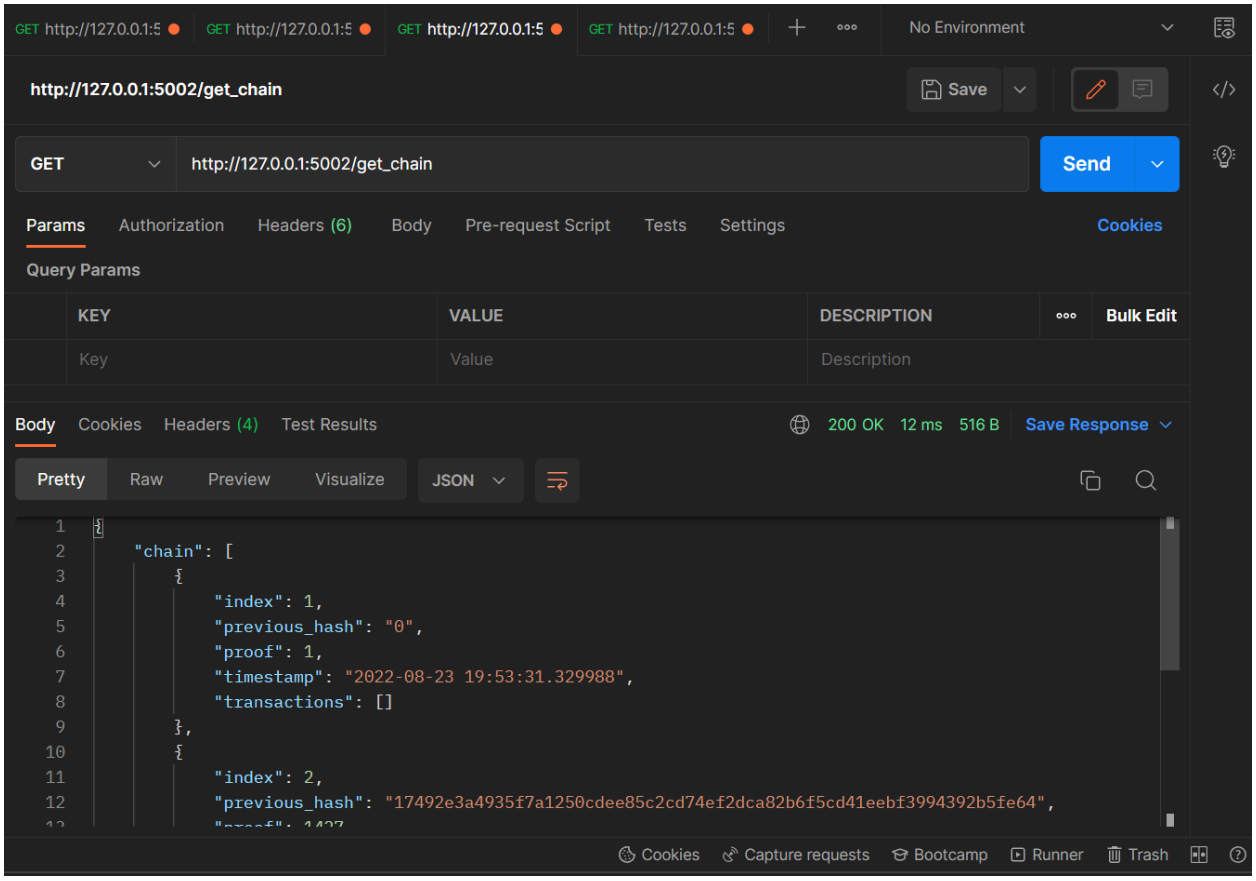
Functioning:

In the Flask application, the nodes are interconnected and thus, when the requests are made in Postman, they are immediately updates for all nodes (users)




GET http://127.0.0.1:5000/g
GET http://127.0.0.1:5001/ge
No Environment
http://127.0.0.1:5001/get_chain
Save
GET
http://127.0.0.1:5001/get_chain
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (4)
Test Results
200 OK
12 ms
271 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"chain": [
{
"index": 1,
"previous_hash": "0",
"proof": 1,
"timestamp": "2022-08-23 19:53:21.662654",
"transactions": []
}
],
"length": 1


GET http://127.0.0.1:5000/g
GET http://127.0.0.1:5001/ge
GET http://127.0.0.1:5002/m
No Environment
http://127.0.0.1:5002/mine_block
Save
GET
http://127.0.0.1:5002/mine_block
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (4)
Test Results
200 OK
37 ms
427 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"index": 2,
"message": "A Block Has Been Mined",
"previous_hash": "17492e3a4935f7a1250cdee85c2cd74ef2dca82b6f5cd41eebf3994392b5fe64",
"proof": 1427,
"timestamp": "2022-08-23 19:55:01.499371",
"transactions": [
{
"amount": 1,
"receiver": "WaWaWeWa",
"sender": "a6f2543e91d84b079e46c8d27fcd35e2"
}
Cookies
Capture requests
Bootcamp
Runner
Trash



Testing:
Features to be tested:

- Validation
- Fault tolerance
- Connection:

Test results:
All test cases mentioned above passed successfully. No defects encountered.

Conclusion:
Thus, this blockchain system has been built and can be used for multiple applications.

References:

1. https://Geeksforgeeks.com
2. https://Wikipedia.com
3. https://flask.palletsprojects.com/en/2.2.x/
4. https://www.postman.com/product/what-is-postman/
5. https://www.udemy.com/