

Blockchain Explorer

Objective:

- Create an application to query block data from the ethereum chain.
- Users should be able to fetch block data using block hash or block number.
- Users should be able to fetch transaction data using transaction hash.
- Users should be able to connect metamask wallets to the website. When metamask is connected data should be fetched from the metamask provider. When it is not connected, data should be fetched via the backend.

Technical Requirements:

- Use any backend, frontend or fullstack framework.
- Create APIs to fetch block and transaction data based on the above parameters.
- Create UI to search data with both metamask and backend APIs integrated.
- Create and share a github repository for the same.
- **Do not use third party APIs like etherscan or moralis to fetch chain data.**

Deliverables:

- Source code with working platform to fulfil above requirements.
- Documentation with concise description of the platform and implementation, what and why you have used to build it and instructions to run the service.

-To Complete the above objectives and the technical requirements we've used the

Following tools and their library:

- *IDE Used: PyCharm*

1) Python

a) Flask

b) Web3

2) Infura

3) Metamask

-Major Points to understand:

1) Web3.py: Web3.py is a Python library for interacting with Ethereum. It's commonly found in decentralized apps (Dapps) to help with sending transactions, interacting with smart contracts, reading block data, and a variety of other use cases. The original API was derived from the Web3.js JavaScript API, but has since evolved toward the needs and creature comforts of Python developers.

2) Ethereum Chain: Ethereum is a decentralized blockchain platform that establishes a peer-to-peer network that securely executes and verifies application code, called smart contracts. Smart contracts allow participants to transact with each other without a trusted central authority. There are roughly 120 million Ethereum chains.

3) Block: A block is a place in a blockchain where information is stored and encrypted. Blocks are identified by long numbers that include encrypted transaction information from previous blocks and new transaction information.

4) Block Hash: The block header is the hash returned from generating a Merkle tree that is below the current difficulty target for the blocks data. The block hash is a value containing the hexadecimal data.

5) Transaction: An Ethereum transaction refers to an action initiated by an externally-owned account, in other words an account managed by a human, not a contract. It is just like a transaction from one's account to another.

6) Transaction Hash: A transaction hash/ID (often abbreviated as tx hash or txn hash) is a unique identifier, similar to a receipt, that serves as proof that a transaction was validated and added to the blockchain. It contains data in the hexadecimal format.

7) Metamask: MetaMask is a free web and mobile crypto wallet that allows users to store and swap cryptocurrencies, interact with the Ethereum blockchain ecosystem, and host a growing array of decentralized applications (dApps). It is one of the most widely used crypto applications in the world.

8) Infura: Infura provides the tools and infrastructure that allow developers to easily take their blockchain application from testing to scaled deployment - with simple, reliable access to Ethereum and IPFS.

9) Flask: A Web Application Framework or simply a Web Framework represents a collection of libraries and modules that enable web application developers to write applications without worrying about low-level details such as protocol, thread management, and so on. Flask is a web framework, it's a Python module that lets you develop web applications easily.

OBJECTIVE-1:

-To Complete the first objective i.e. to create an application to query block data from the Ethereum chain

We have written thus code to connect to the Ethereum chain using the *Infura*:

```
# Import Essential lib #web3
from web3 import Web3
from web3.middleware import geth_poa_middleware

# Connect to an Ethereum node
w3 =
Web3(Web3.HTTPProvider('https://celo-mainnet.infura.io/v3/c0675c5a775e421db5430b57358c8b52'))
w3.middleware_onion.inject(geth_poa_middleware, layer=0)
print(w3)
print(w3.isConnected())

block_number = w3.eth.blockNumber
# Get the block data for the latest block
Block_data = w3.eth.get_block(block_number)
print("Block Number: ", block_number)
# print("Block Hash: ", block['hash'])
print("Block Timestamp: ", block['timestamp'])
print("Block Transactions: ", block['transactions'])
print("Block_data")
```

OUTPUT:



```
blockData_fromEthChain
C:\Users\adity\anaconda3\python.exe "C:\Users\adity\PycharmProjects\pythonProject\block data from eth chain\blockData_fromEthChain.py"
<web3.main.Web3 object at 0x000001DF4F896640>
True
Block Number: 17396849
Block Hash: b'\xcd6S\xa1(m\xcd\xbc\xd7\x15\xc5\\\x8f\xde\xbd\x0c\xf8\xdb\xbe\n\xc3\xa5)\x1do\xea:Ny;JE6'
Block Timestamp: 1674652303
Block Transactions: [HexBytes('0xe61c7dc5bd038863ed541a369743eb07f9886fc35207a61ae4afb9862651512a'), HexBytes
('0xbdb06c3601b5081890c2a9f8e49dda592bd063622f3400cadf2fcfaa8dd9ef99'), HexBytes('0x714ee10d32ca768e88f143043e268c6e5c58d0261202f01d9e4a5d23605e9e5f'),
HexBytes('0xea6bb31a96e304240039ee6e8bcd152df8f0edcb79df666bd32b9a1b7fb6ff66'), HexBytes
('0xbcb5df49db64baa1d3eb6bdd418aee180e5448b060b8c5a1a8df50a5f48723f0'), HexBytes('0x760ee6c57f3208c66013c6d6b3f7355dcf21e3755cafae764e275b48afc7625d'),
HexBytes('0xf00d4fd2243361a1a57fb12a5338435ee258e440ff4e7000d9d627df01ddff0b'), HexBytes
('0x9c56ddf140c9572a6b55f0b0c10f00f0e1655ba0c292b4638fb3f47f89ef2bc3'), HexBytes('0x256199ca9d57a2f50541b3087446cbc427d056248d7038037e7a716de26fdb52'),
HexBytes('0x795057403771a9d62ef4e130f6aeecd4a97f474404934fd7201c2a87b3d9651a'), HexBytes
('0x1ca609f35299f5d085626201f715649b57626cc0c83958271bb5bca3e2893680'), HexBytes('0x3a900b5ea1b293281fc10837654cd3c327014e308235c6b8fee1fb517dd4a9fa'),
HexBytes('0x72ef33667ed4a003adb2384e1f4c491a18b4d3da1400e5d95a210a3efff262afb'), HexBytes
('0xc3426ab895cf6bb22dd2c2c8a1449adbc00ae9253f30461a146b9f2f10d1dc'), HexBytes('0x776035880bf6271208d898e5b28622d1e2b652b68900ce04cc1ffc85b9812762'),
HexBytes('0xdb327a5d49374c8920b5b8358f8441fb729134ac600ca250dd55af7284aca284'), HexBytes
('0xcc851dd25288e9f79db541d884022500776811a0ed3c2d2b21668bdc6da1046'), HexBytes('0x81932a7439efb9c9c8c6198479dc8b0bd1310123193a473135833b65905fd6e3'),
HexBytes('0x2ceafe449555d77ac7ce6f4af5487b6e7a328e6f5de91f342f6399a7b52a70e4'), HexBytes
('0x6e7c3cea597767ee67a0a86fae29607fe01ad1629980ee80daf51c613f472f73'), HexBytes('0x87d13aa3d56593bf06c13f94f5f10bb5bf7173f3351b6007a05a3af169ed8a68'),
HexBytes('0xaa48c2b2de6171932d7e156d32dc479c37e6347de1af8b2ead71e4dcd2e7644f'), HexBytes
('0xed7503b69257709cf61dbb46d74f8856411aa203e22bb12e985e76f46305f420'), HexBytes('0x71fe439087c58eaa9208638e8217886a486d268843ee81cde3d64f7f7f80e90c')]
```

Process finished with exit code 0

This code connects to the Ethereum mainnet using infura, a service that allows you to access the Ethereum without having to run your own code.

You can also use the web3.py library to connect to a local node or a testnet, you can use the following code to connect to a local node.

```
"w3 = Web3(Web3.HTTPProvider("http://127.0.0.1:8545"))"
```

OBJECTIVE-2

-To complete the second objective we have written a code snippet to fetch the block data using the block number and the block hash:

```
from web3 import Web3
from flask import Flask, jsonify
from web3.middleware import geth_poa_middleware

#Getting block data using block number & block Hash
def get_block_data_num(block_number):
    w3 = Web3(Web3.HTTPProvider('https://celo-mainnet.infura.io/v3/c0675c5a775e421db5430b57358c8b52'))
    w3.middleware_onion.inject(geth_poa_middleware, layer=0)

    Block_data_using_num = w3.eth.get_block(block_number)  #Using Block Number
    print(Block_data_using_num)

block_number = int(input("Enter Block Number: "))
get_block_data_num(block_number)

print("¥¥n")

def get_block_data_hash(block_hash):
    w3 = Web3(Web3.HTTPProvider('https://celo-mainnet.infura.io/v3/c0675c5a775e421db5430b57358c8b52'))
    w3.middleware_onion.inject(geth_poa_middleware, layer=0)

    #Hash Valuse:      '0x55a2d76a8c181f5ff01efc382aea84a8e886172999459c0f83bd1ba6f57a8d6e'
    Block_data_using_hash = w3.eth.get_block(block_hash)  #Using Block Hash
    print(Block_data_using_hash)

block_hash = input("Enter Block Hash: ")
get_block_data_hash(block_hash)
```

OUTPUT:


```

from web3 import Web3
from flask import Flask, jsonify
from flask_cors import CORS
from web3.middleware import geth_poa_middleware

#Getting Transaction data using transaction hash:
#0x1287bce9d76712f6410c585e1921056f1abfc98c9bb8b57680873de561160234'

def get_transaction_data(transaction_hash):
    w3 = Web3(Web3.HTTPProvider('https://celo-mainnet.infura.io/v3/c0675c5a775e421db5430b57358c8b52'))
    w3.middleware_onion.inject(geth_poa_middleware, layer=0)
    Transaction_data = w3.eth.get_transaction(transaction_hash) #Using Transaction Hash
    print(Transaction_data)

transaction_hash = input("Enter Transaction Hash: ")
get_transaction_data(transaction_hash)

```

OUTPUT:

```

transactionData_usingTransactionHash
C:\Users\adity\anaconda3\python.exe "C:\Users\adity\PycharmProjects\pythonProject\block & transaction\transactionData_usingTransactionHash.py"
Enter Transaction Hash: 0x1287bce9d76712f6410c585e1921056f1abfc98c9bb8b57680873de561160234
AttributeDict({'blockHash': HexBytes('0xd396769949b93cd596bd9dfdd3a3df0405881ba9542177a7156274df214e530f'), 'blockNumber': 17368381, 'ethCompatible':
False, 'feeCurrency': '0x765de816845861e75a25fca122bb6898b8b1282a', 'from': '0xb0C220727a809eD048AC5C3Fd87b8d4cC3A49DAb0', 'gas': 623611, 'gasPrice':
15000000000, 'gatewayFee': '0x0', 'gatewayFeeRecipient': None, 'hash': HexBytes('0x1287bce9d76712f6410c585e1921056f1abfc98c9bb8b57680873de561160234'),
'input': '0xc9f9d0b5f', 'nonce': 39, 'r': HexBytes('0x72ffe875e67fb4d2a9dadac1955db28082cdd4b0f58990d2672db007a7a790d'), 's': HexBytes
('0x5aed70b66f986dd0289799550323828449bad890efc9bd94067e9fd2edc399d1'), 'to': '0x1C51657af2ceBA3D5492bA0c5A17E562F7ba6593', 'transactionIndex': 0,
'type': '0x0', 'v': 84476, 'value': 0})

Process finished with exit code 0
|

```

OBJECTIVE-4

-To connect metamask wallets to the website we have created an UI and a backend using the flask.

When metamask is connected data will be fetched from the metamask provider as below code worked:

Metamask.py:

```

from flask import Flask, render_template

# Flask App

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

```

The above code connect the flask application to the html page that we have created to fetch the data from the metamask below:

Index.html:

```
<html>
  <head>
    <title>MetaMask Connection Page</title>

  </head>
  <body>
    <center>
      <h1>Connect Your MetaMask Wallet Here!</h1>
      <button id='connectWallet' onclick="">Connect Wallet</button>
      <button id='getBalance' onclick="checkBalance()">Get Balance of Wallet</button>

      <p id="walletAddress"></p>
      <p id="walletBalance"></p>

      <script type="text/javascript">
        window.userWalletAddress = null
        const connectWallet = document.getElementById('connectWallet')
        const walletAddress = document.getElementById('walletAddress')
        const walletBalance = document.getElementById('walletBalance')

        function checkInstalled() {
          if (typeof window.ethereum == 'undefined') {
            connectWallet.innerText = 'MetaMask isnt installed, please install it'
            connectWallet.classList.remove()
            connectWallet.classList.add()
            return false
          }
          connectWallet.addEventListener('click', connectWalletwithMetaMask)
        }

        async function connectWalletwithMetaMask() {
          const accounts = await window.ethereum.request({ method: 'eth_requestAccounts' })
          .catch((e) => {
            console.error(e.message)
            return
          })

          if (!accounts) { return }

          window.userWalletAddress = accounts[0]
          walletAddress.innerText = window.userWalletAddress

          connectWallet.innerText = 'Sign Out'
          connectWallet.removeEventListener('click', connectWalletwithMetaMask)
          setTimeout(() => {
            connectWallet.addEventListener('click', signOutOfMetaMask)
          }, 200)

        }

        function signOutOfMetaMask() {
          window.userwalletAddress = null
          walletAddress.innerText = ''
          . . . . .
        }
      </script>
    </center>
  </body>
</html>
```


-The above code contains the code which collects the present in the metamask wallet and shows the data on that HTML page that we have created.

OUTPUT:

Connect Your MetaMask Wallet Here!

Connect Your MetaMask Wallet Here!

0x40ffc058c7e30fc3d42cc92282db9bdc5d77d25f

0

TECHNICAL REQUIREMENTS:

REQUIREMENT -1:

-Use any backend, frontend or full-stack framework.

We have used:

- 1) Python-Flask frame work for back-end.
- 2) HTML, CSS, Java-Script for front-end.

REQUIREMENT -2:

-Creating APIs to fetch block and transaction data based on the above parameters:

We have used the Python-Flask framework to create the APIs to fetch Block Data & Transaction data.

OUTPUT:


```

import json
# from attrdict import AttrDict
from web3 import Web3
from flask import Flask, jsonify
from flask_cors import CORS
from web3.middleware import geth_poa_middleware

app = Flask(__name__)
# CORS(app)
CORS(app, resources={r"/api/*": {"origins": "*"}}, headers='Content-Type')

#Getting block data using block number
@app.route('/api/block/number/<int:block_number>', methods=['GET'])
def fetch_block_data_num(block_number):
    try:
        w3 =
Web3(Web3.HTTPProvider('https://celo-mainnet.infura.io/v3/c0675c5a775e421db5430b57358c8
b52'))
        w3.middleware_onion.inject(geth_poa_middleware, layer=0)
        Block_data = w3.eth.getBlock(block_number)

        return json.dumps(Block_data, default=str)
    except:
        return json.dumps("Please Enter Block Number")

print("\n")

#Getting block data using block Hash
# '0x55a2d76a8c181f5ff01efc382aea84a8e886172999459c0f83bd1ba6f57a8d6e'

@app.route('/api/block/hash/<block_number>', methods=['GET'])
def fetch_block_data_hash(block_number):
    try:
        w3 =
Web3(Web3.HTTPProvider('https://celo-mainnet.infura.io/v3/c0675c5a775e421db5430b57358c8
b52'))
        w3.middleware_onion.inject(geth_poa_middleware, layer=0)
        Block_data = w3.eth.getBlock(block_number)

        return json.dumps(Block_data, default=str)
    except:
        return json.dumps("Please Enter Block Hash")

print("\n")

# Getting Transaction data using transaction hash:
# '0x1287bce9d76712f6410c585e1921056f1abfc98c9bb8b57680873de561160234'
@app.route('/api/block/transaction_hash/<transaction_hash>', methods=['GET'])
def fetch_transaction_data(transaction_hash):
    try:
        w3 =
Web3(Web3.HTTPProvider('https://celo-mainnet.infura.io/v3/c0675c5a775e421db5430b57358c8
b52'))
        w3.middleware_onion.inject(geth_poa_middleware, layer=0)
        Transaction_data = w3.eth.get_transaction(transaction_hash)

        return json.dumps(Transaction_data, default=str)
    except:
        return json.dumps("Please Enter Transaction Hash")

if __name__ == '__main__':
    app.run(debug=True, port=5000)

```

OUTPUT:

```
blockData_usingApiWebServer x
C:\Users\adity\anaconda3\python.exe "C:\Users\adity\PycharmProjects\pythonProject\block & transaction\blockData_usingApiWebServer.py"

* Serving Flask app "blockData_usingApiWebServer" (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: on
* Restarting with windowsapi reloader

* Debugger is active!
* Debugger PIN: 333-197-337
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

EQUIREMENT -3:

- Created UI to search data with both meta-mask and backend APIs integrated.

We have created the UI using the HTML, CSS & Java-Script to search the data.

Blockchain Explorer

Block Number

Block Hash

Transaction Hash

SUBMIT

You are fetching Block Data using Block Number

CREDIT: ADITYA CHOUHAN

OUTPUT

REQUIREMENT -4:

- Created and shared a GitHub repository for the same.

-Aditya Chouhan

Date: 27-01-2023

```
AttributeDict({'baseFeePerGas': 100000000, 'epochSnarkData': None,  
  'proofOfAuthorityData':  
    HexBytes('0xd983010000846765746889676f312e31332e3130856c696e757800000000  
      000f8c2c0c80b841571b179ffea3caadd94367ae6c8bb19baac9c405bf60f31a40bd4fff3b  
      24e74c8707c77bc3b0b47fbfb05a4e2423d475040c8fad69278a220473bc5dc6ba1eb21540  
      1f83c8901c5a374de3fd9f7bf5b0825ae053f821b9d2bd451f7b3f3ac5c6204ab6e983df  
      4b2e2d44617aa398ec25e95c827de6cdc87a2a04908f78080f83c8901efeffffffffffffb04  
      2c1e8b45dd4eece539f32c4e4a4f0691c20e2c1ce93d07ba076a1ec050860ec2b74139b1  
      40c8ac7dd9843d2638080b'),  
    'gasLimit': 10000000, 'gasUsed': 0, 'hash':  
    HexBytes('0x55a2d76a8c181f5ff01efc382aea84a8e886172999459cf0f83bd1ba6f57a8d  
      e'), 'logsBloom':  
    HexBytes('0x0000000000000000000000000000000000000000000000000000000000000000  
      000000000000000000000000000000000000000000000000000000000000000000000000  
      000000000000000000000000000000000000000000000000000000000000000000000000  
      040000000000000000000000000000000000000000000000000000000000000000000000  
      000000000000000000000000000000000000000000000000000000000000000000000000  
      000000000000000000000000000000000000000000000000000000000000000000000000  
      000000000000000000000000000000000000000000000000000000000000000000000000  
      000000000000000000000000000000000000000000000000000000000000000000000000  
      000000000000000000000000000000000000000000000000000000000000000000000000  
      000000000000000000000000000000000000000000000000000000000000000000000000'), 'miner':  
    'OxF27bb4eABC4400A1ABE9D80D7537ab0Ef1B058BF', 'number': 12345, 'parentHash':  
    HexBytes('0xa8a692f30daf7aba5246f7662adca1bc8c2e231d999d832a4a1b87ffc0131d  
      2'), 'randomness': AttributeDict({'committed':  
    '0x61643b7e91f8810d335fa7ca47de5b05efd5db235876c66f1df876690ace27d',  
    'revealed':  
    '0xc253942f5407a7fc866479e0b17fb5368331267da124a55df0a26ef019fc3bc)'),  
    'receiptsRoot':  
    HexBytes('0x42e3b121d833b692e4d52b4187e5c12dca2898d788bdc3d3641a6ccfef6fa
```

