OEIT1-Blockchain Technology and Applications
AY:2023-2024

Lab1A:Tracking the Blockchain Transactions using API

Date: 29/01/24
UCID: 2021300101
Sem: 6

Student Name: Adwait Purao
Branch: Comps B

Objective: To explore the blockchain transactions and API

Outcomes: After completing the lab student will be able to
[1] Describe the Block structure in Blockchain
[2] Describe the a block, transactions, hash, cryptocurrency (BTC, ETH), Gas concept, Gas price, Gas used, Gas limit
[3] Use Blockchain API to track the transactions in Blockchain
[4] Perform the statistical analysis of Blockchain transactions.

Refer to Github Classroom

Step-1: Use VS code and push the code to Github

Step-2: Provide pseudo code

```
IMPORT requests
IMPORT pandas
IMPORT matplotlib.pyplot
IMPORT DateFormatter
IMPORT YearLocator

DEFINE api_key
DEFINE address

DEFINE endpoint using api_key and address

SEND GET request to endpoint

IF response status code EQUALS 200
```

PARSE JSON response
IF data status EQUALS '1'
EXTRACT transactions from data result
CREATE DataFrame tx_df to store transactions
CONVERT 'timeStamp' to datetime format in tx_df
CONVERT 'value' from wei to Ether in tx_df
CONVERT 'gasUsed' to float in tx_df
CALCULATE gasPaid = gasUsed * gasPrice in tx_df
PLOT Account Time vs Ether(ETH) Value
SET figure size
PLOT timeStamp vs value with markers
SET title, xlabel, ylabel, and grid
CUSTOMIZE x-axis tick format to display years
SET tick locator to include all years
SHOW plot
PLOT Account Time vs Gas paid (in ETH)
SET figure size
PLOT timeStamp vs gasPaid with markers in red
SET title, xlabel, ylabel, and grid
CUSTOMIZE x-axis tick format to display years
SET tick locator to include all years
SHOW plot
ELSE
PRINT 'No transactions found for the address'
ELSE
PRINT 'Failed to fetch data from the API:', response status code

Code 1:

```
import requests
import json

def get_transaction_details(tx_hash, api_key):
    url =
f"https://api.etherscan.io/api?module=proxy&action=eth_getTransactionByHas
h&txhash={tx_hash}&apikey={api_key}"
    response = requests.get(url)
    data = response.json()
    return data

def print_transaction_details(transaction):
    print(json.dumps(transaction, indent=4))

# Replace with your transaction hash and API key
```

```
transaction_hash =
'0x0f0f23f081fb91b47334a678bda61be2b459b4f438762436e9a64d87791a257c'
api_key = 'YOUR_API_KEY_HERE'

transaction_details = get_transaction_details(transaction_hash, api_key)
print_transaction_details(transaction_details)
```

Output:

```
itlab@itlab-OptiPlex-3000:~/Desktop/Adwait BCT$ /bin/python3 "/home/itlab/Desktop/Adwait BCT/exp1.py"
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": {
        "blockHash": "0xce888ab3466366fa2f0e6a96637fcf2aecd89b3b5306a2bbf433d98306a4a2ac",
        "blockNumber": "0x1239d0e",
        "from": "0x4838b106fce9647bdf1e7877bf73ce8b0bad5f97",
        "gas": "0x6ac1",
        "gasPrice": "0x292976f4d",
        "maxFeePerGas": "0x292976f4d",
        "maxPriorityFeePerGas": "0x0",
        "hash": "0x0f0f23f081fb91b47334a678bda61be2b459b4f438762436e9a64d87791a257c",
        "input": "0x",
        "nonce": "0x2b83b",
        "to": "0xba1951df0c0a52af23857c5ab48b4c43a57e7ed1",
        "transactionIndex": "0xa4",
        "value": "0x3f645de6372ea0",
        "type": "0x2",
        "accessList": [],
        "chainId": "0x1",
        "v": "0x0",
        "r": "0x3c19626cd28a70734dcdc9639d7be9382cbb75e334c542dba9e7b0bd7a53afdd",
        "s": "0x9ce1b7a5267be5c0af5544ca20010003c608fc532e847cec344f853bdf7d01e",
        "yParity": "0x0"
    }
}
```

Code 2:

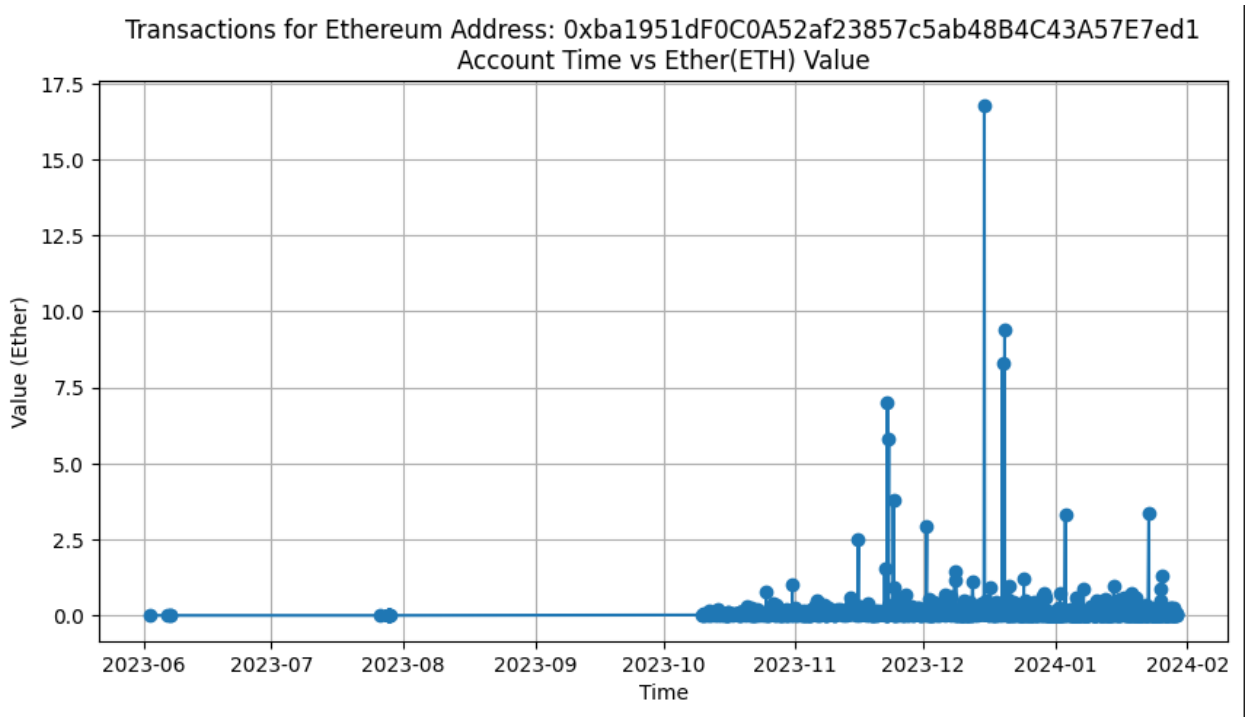
```
import requests
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter, YearLocator

# Your Etherscan API key
api_key = 'YOUR_API_KEY_HERE'

# Example Ethereum address
address = '0xba1951dF0C0A52af23857c5ab48B4C43A57E7ed1'

# Example API endpoint for getting transactions
endpoint =
f'https://api.etherscan.io/api?module=account&action=txlist&address={addre
ss}&apikey={api_key}'

# Make a GET request to the API
response = requests.get(endpoint)
```

```
# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Parse the JSON response
    data = response.json()

    # Check if the response contains transactions
    if data['status'] == '1':
        # Extract the list of transactions from the response
        transactions = data['result']

        # Create a DataFrame to store the transaction data
        tx_df = pd.DataFrame(transactions)

        # Convert 'timeStamp' to datetime format
        tx_df['timeStamp'] = pd.to_datetime(tx_df['timeStamp'], unit='s')

        # Convert 'value' from wei to Ether
        tx_df['value'] = tx_df['value'].astype(float) / 10**18

        # Convert 'gasUsed' to float
        tx_df['gasUsed'] = tx_df['gasUsed'].astype(float)

        # Calculate gas paid in Ether (gasUsed * gasPrice)
        tx_df['gasPrice'] = tx_df['gasPrice'].astype(float) / 10**18
        tx_df['gasPaid'] = tx_df['gasUsed'] * tx_df['gasPrice']

        # Plot: Account Time vs Ether(ETH) Value
        plt.figure(figsize=(10, 5))
        plt.plot(tx_df['timeStamp'], tx_df['value'], marker='o')
        plt.title(f'Transactions for Ethereum Address: {address}\nAccount
Time vs Ether(ETH) Value')
        plt.xlabel('Time')
        plt.ylabel('Value (Ether)')
        plt.grid(True)

        # Customize x-axis tick format to display years
        date_format = DateFormatter("%Y")
        plt.gca().xaxis.set_major_formatter(date_format)
```

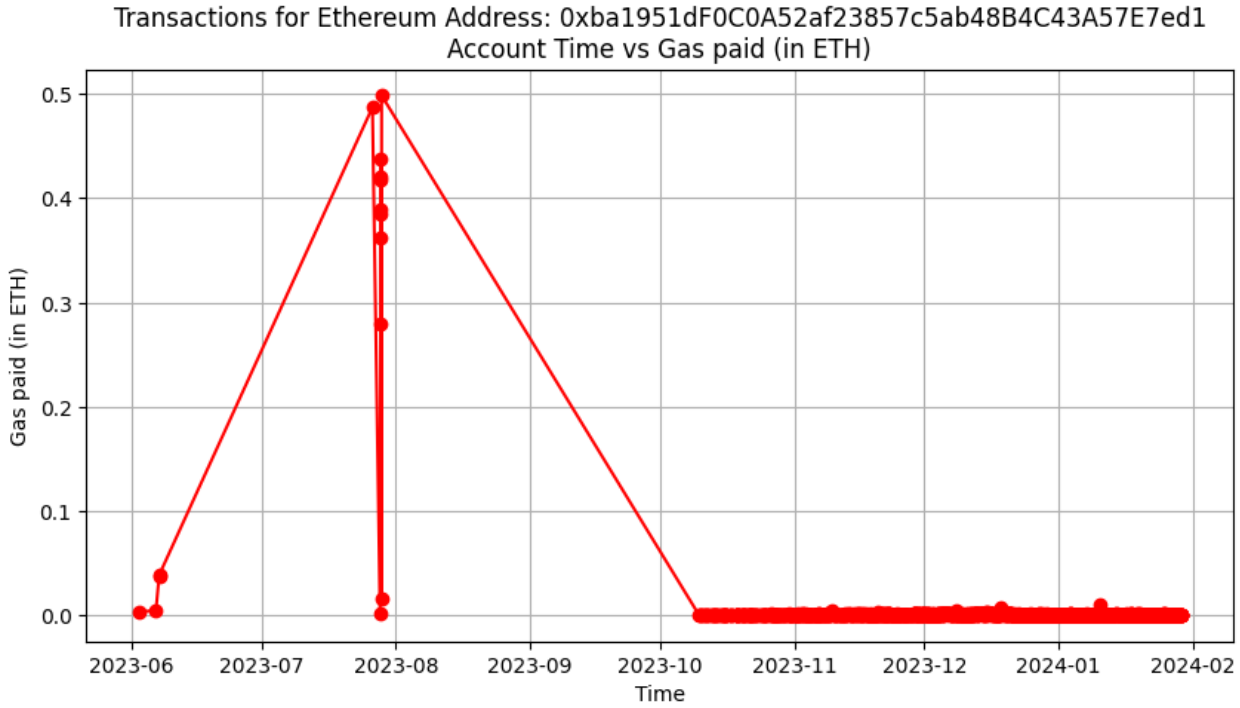
```
        # Set the tick locator to include all years
        plt.gca().xaxis.set_major_locator(YearLocator())

        plt.show()

        # Plot: Account Time vs Gas paid (in ETH)
        plt.figure(figsize=(10, 5))
        plt.plot(tx_df['timeStamp'], tx_df['gasPaid'], marker='o',
color='r')
        plt.title(f'Transactions for Ethereum Address: {address}\nAccount
Time vs Gas paid (in ETH)')
        plt.xlabel('Time')
        plt.ylabel('Gas paid (in ETH)')
        plt.grid(True)

        # Customize x-axis tick format to display years
        plt.gca().xaxis.set_major_formatter(date_format)

        # Set the tick locator to include all years
        plt.gca().xaxis.set_major_locator(YearLocator())

        plt.show()
    else:
        print('No transactions found for the address')
else:
    print('Failed to fetch data from the API:', response.status_code)
```

Step-3: Add plot with caption

[i]Account Time vs Ether(ETH) Value



[ii] Account Time vs Gas paid (in ETH)



Conclusion:

In conclusion, this experiment offered a comprehensive understanding of blockchain fundamentals, including block structure, transactions, and key concepts like hashing and gas in Ethereum transactions. Participants utilized blockchain APIs to track transactions and

performed statistical analysis. Additionally, they gained practical experience in accessing and analyzing blockchain data, culminating in the visualization of transaction trends. This hands-on approach provided valuable insights into blockchain's real-world applications and its potential impact across diverse sectors.