

**OEIT1-Blockchain Technology and Applications**

**AY:2023-2024**

**Lab1A:Tracking the Blockchain Transactions using API**

Date: 29/01/24 Student Name: Adwait Purao

UCID: 2021300101 Branch: Comps B

Sem: 6

**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\_getTransactionByHash&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:**



**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={address}&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:

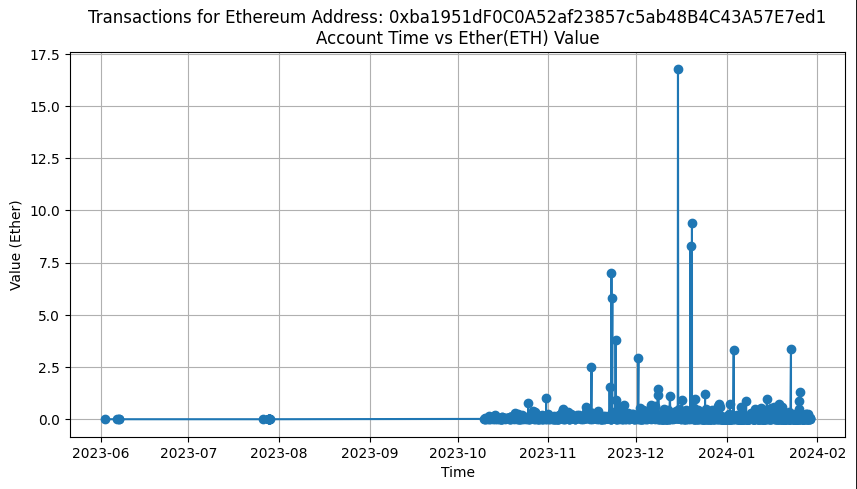
print('No transactions found for the address')

else:

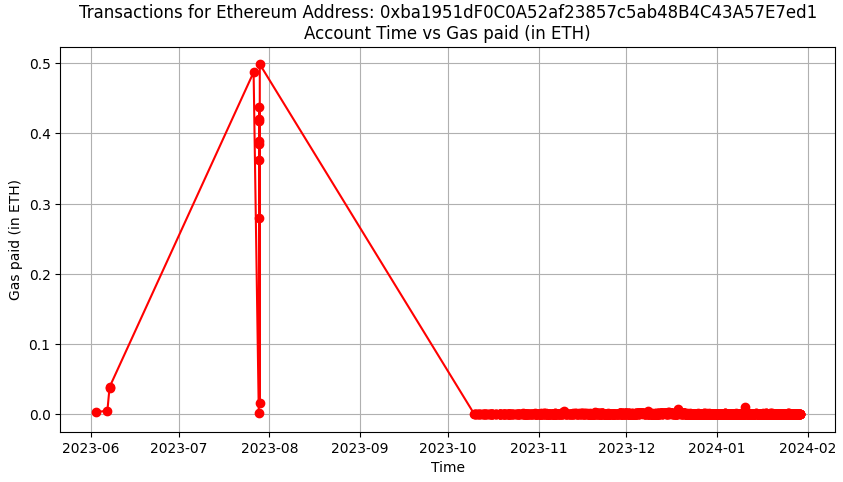
print('Failed to fetch data from the API:', response.status\_code)

**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.