CASE STUDY : Analysis of System Resource Usage

Implementation

main.py

```
import subprocess
import schedule
import time

def job():
    subprocess.run(["python", "collect_data.py"])

schedule.every(2).seconds.do(job)

while True:
    schedule.run_pending()
    time.sleep(1)
```

collect_data.py

```
import os
import psutil
import csv
import datetime
import logging

timestamp = datetime.datetime.now()

cpu_usage = psutil.cpu_percent(1)
process = psutil.Process(os.getpid())
mem_usage = process.memory_percent()
disk_usage = psutil.disk_usage('/').percent

row = {'timestamp': timestamp, 'cpu_usage': cpu_usage, 'memory_usage': mem_usage, 'disk_usage':
disk_usage}
with open('system_resource.csv', 'a', newline='') as f:
    writer = csv.DictWriter(f, fieldnames=row.keys())
    writer.writerow(row)

logging.basicConfig(filename='system_resource.log', level=logging.DEBUG)
logging.debug(f"{timestamp} | {cpu_usage} | {mem_usage} |{disk_usage}")
```

statistics.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Read the data
df = pd.read_csv("system_resource.csv", names=['timestamp', 'cpu_usage', 'memory_usage',
'disk_usage'])

# Overview of data
print(df.info())

timestamp = df['timestamp']
cpu_usage = df['cpu_usage']
memory_usage = df['memory_usage']
disk_usage = df['disk_usage']

mean_cpu = np.mean(cpu_usage)
mean_mem = np.mean(memory_usage)
mean_disk = np.mean(disk_usage)

median_cpu = np.median(cpu_usage)
median_mem = np.median(memory_usage)
median_disk = np.median(disk_usage)

std_cpu = np.std(cpu_usage)
std_mem = np.std(memory_usage)
std_disk = np.std(disk_usage)

print("\nStatistics of CPU Usage :: ")
print(f"Mean: {mean_cpu}, Median: {median_cpu}, Standard Deviation: {std_cpu}")

print("\nStatistics of Memory Usage :: ")
print(f"Mean: {mean_mem}, Median: {median_mem}, Standard Deviation: {std_mem}")

print("\nStatistics of Disk Usage :: ")
print(f"Mean: {mean_disk}, Median: {median_disk}, Standard Deviation: {std_disk}")


# Plot the data
plt.plot(timestamp, cpu_usage, label="CPU Usage")
plt.xlabel("Timestamp")
plt.ylabel("CPU Usage")
plt.title("CPU Usage over Time")
plt.legend()
plt.show()
```

```
plt.plot(timestamp, memory_usage, label="Memory Usage")
plt.xlabel("Timestamp")
plt.ylabel("Memory Usage")
plt.title("Memory Usage over Time")
plt.legend()
plt.show()

plt.plot(timestamp, disk_usage, label="Disk Usage")
plt.xlabel("Timestamp")
plt.ylabel("Disk Usage")
plt.title("Disk Usage over Time")
plt.legend()
plt.show()
```

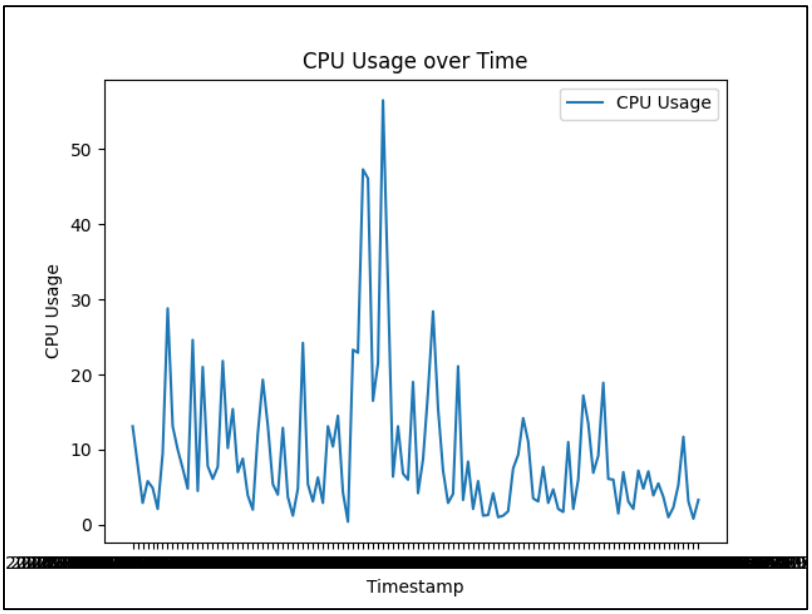
Output

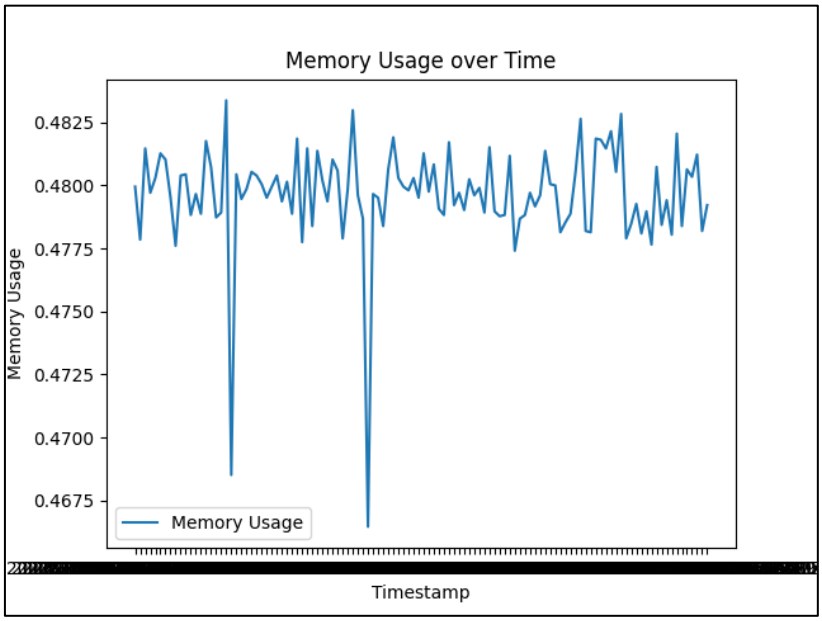
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114 entries, 0 to 113
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   timestamp     114 non-null    object
 1   cpu_usage     114 non-null    float64
 2   memory_usage  114 non-null    float64
 3   disk_usage    114 non-null    float64
dtypes: float64(3), object(1)
memory usage: 3.7+ KB
None
```

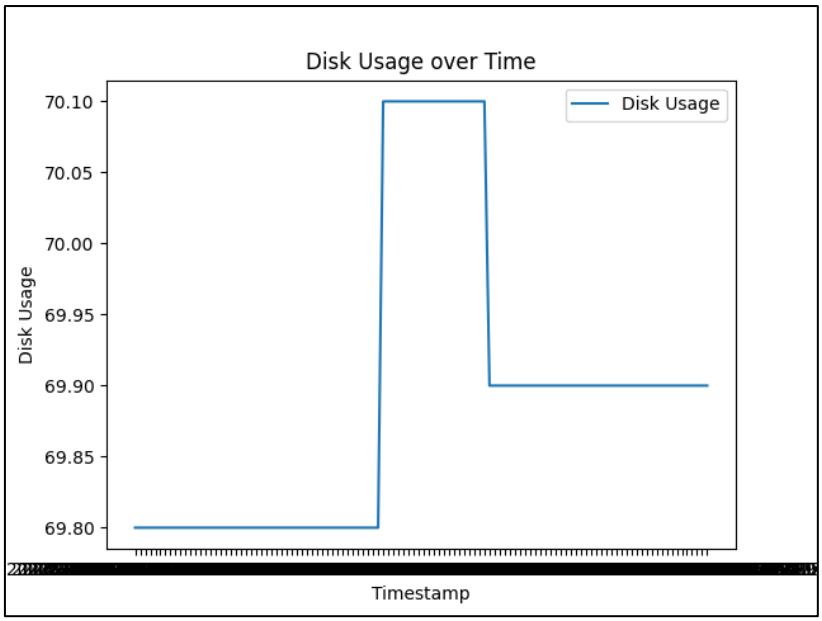
```
Statistics of CPU Usage ::
    Mean: 9.718421052631578
    Median: 6.6
    Standard Deviation: 9.595142829993613

Statistics of Memory Usage ::
    Mean: 0.47965077735024114
    Median: 0.4797060774009861
    Standard Deviation: 0.0020756176272882874

Statistics of Disk Usage ::
    Mean: 69.89385964912279
    Median: 69.9
    Standard Deviation: 0.10783766854755089
```


CPU Usage over Time
CPU Usage
CPU Usage
0
10
20
30
40
50
Timestamp


Memory Usage over Time
Memory Usage
Memory Usage
0.4675
0.4700
0.4725
0.4750
0.4775
0.4800
0.4825
Timestamp


Disk Usage over Time
Disk Usage
Disk Usage
69.80
69.85
69.90
69.95
70.00
70.05
70.10
Timestamp

```
DEBUG:root:2023-11-06 18:21:47.981133 | 13.1 | 0.4799504264624169 |69.8
DEBUG:root:2023-11-06 18:21:52.148757 | 8.0 | 0.4778490245341119 |69.8
DEBUG:root:2023-11-06 18:21:56.549403 | 2.9 | 0.48146539064328797 |69.8
DEBUG:root:2023-11-06 18:22:01.139351 | 5.8 | 0.4797060774009861 |69.8
DEBUG:root:2023-11-06 18:22:05.411684 | 4.9 | 0.4802925151484201 |69.8
DEBUG:root:2023-11-06 18:22:10.142094 | 2.1 | 0.4812699113941434 |69.8
DEBUG:root:2023-11-06 18:22:14.647585 | 9.5 | 0.48102556233271254 |69.8
DEBUG:root:2023-11-06 18:22:18.957159 | 28.8 | 0.47951059815184144 |69.8
DEBUG:root:2023-11-06 18:22:23.180540 | 13.1 | 0.47760467547268104 |69.8
DEBUG:root:2023-11-06 18:22:27.389990 | 10.0 | 0.48039025477299235 |69.8
DEBUG:root:2023-11-06 18:22:31.513353 | 7.5 | 0.4804391245852786 |69.8
DEBUG:root:2023-11-06 18:22:35.909260 | 4.8 | 0.47882642077983517 |69.8
DEBUG:root:2023-11-06 18:22:39.940022 | 24.6 | 0.47965720758869995 |69.8
DEBUG:root:2023-11-06 18:22:44.245925 | 4.5 | 0.4788752905921213 |69.8
DEBUG:root:2023-11-06 18:22:47.933984 | 21.0 | 0.481758609517005 |69.8
```

	C1	C2	C3	C4
1	2023-11-06 18:21:47.981133	13.1	0.4799504264624169	69.8
2	2023-11-06 18:21:52.148757	8.0	0.4778490245341119	69.8
3	2023-11-06 18:21:56.549403	2.9	0.48146539064328797	69.8
4	2023-11-06 18:22:01.139351	5.8	0.4797060774009861	69.8
5	2023-11-06 18:22:05.411684	4.9	0.4802925151484201	69.8
6	2023-11-06 18:22:10.142094	2.1	0.4812699113941434	69.8
7	2023-11-06 18:22:14.647585	9.5	0.48102556233271254	69.8
8	2023-11-06 18:22:18.957159	28.8	0.47951059815184144	69.8
9	2023-11-06 18:22:23.180540	13.1	0.47760467547268104	69.8
10	2023-11-06 18:22:27.389990	10.0	0.48039025477299235	69.8
11	2023-11-06 18:22:31.513353	7.5	0.4804391245852786	69.8
12	2023-11-06 18:22:35.909260	4.8	0.47882642077983517	69.8
13	2023-11-06 18:22:39.940022	24.6	0.47965720758869995	69.8
14	2023-11-06 18:22:44.245925	4.5	0.4788752905921213	69.8
15	2023-11-06 18:22:47.933984	21.0	0.481758609517005	69.8
16	2023-11-06 18:22:52.289804	7.8	0.48073234345899557	69.8
17	2023-11-06 18:22:56.454984	6.1	0.4787286811552628	69.8