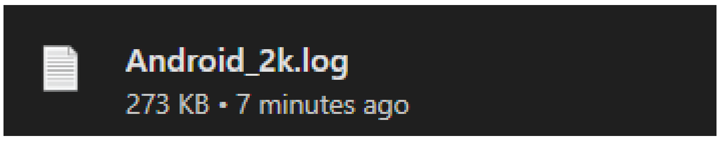
Name: Hannah Emad

ID:2205123

Perform Log File Analysis Using Bash

1. Download a Log File

- A real-world log file (`Android_2k.log`) was downloaded from a public dataset repository such as GitHub LogHub



Android_2k.log
273 KB • 7 minutes ago

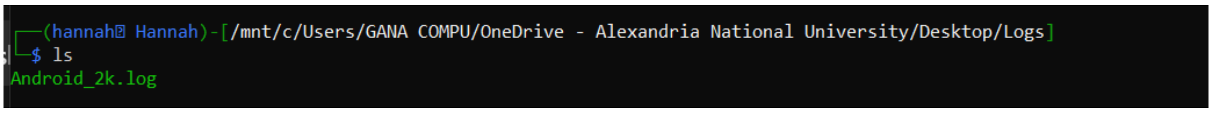
---

2. Prepare the Environment

- Opened a terminal window on a WSL environment.

- Navigated to the directory where the log file is saved.

cd "/mnt/c/Users/GANA COMPU/OneDrive - Alexandria National University/Desktop/Logs"



```
┌──(hannah㉿ Hannah)-[/mnt/c/Users/GANA COMPU/OneDrive - Alexandria National University/Desktop/Logs]
└─$ ls
Android_2k.log
```

---

3. Create the Bash Script

- Write the following command to open the nano text editor and create the parsing script:

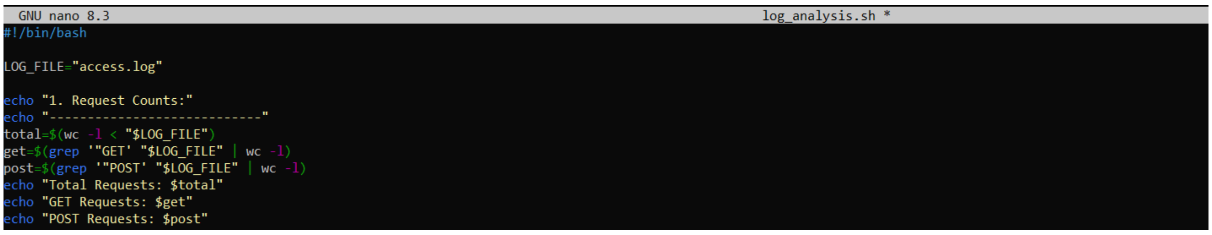
nano log_analysis.sh

We wrote a Bash script that:

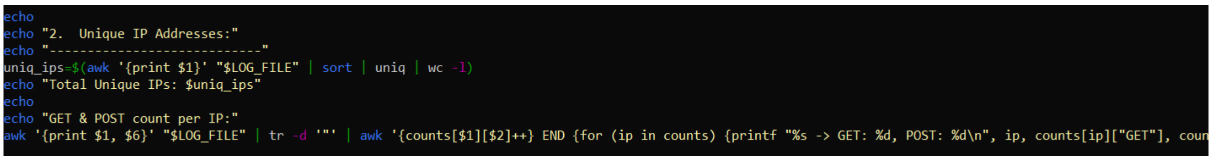
- Counts total, GET, and POST requests.
- Identifies unique IP addresses and request counts per IP.
- Counts failed requests (4xx and 5xx).
- Calculates failure percentage.

- Detects the most active IP.
- Calculates average daily requests.
- Shows hourly request distribution.
- Breaks down HTTP status codes.
- Finds the most active users per request method.
- Detects patterns in failure requests (by hour and day).

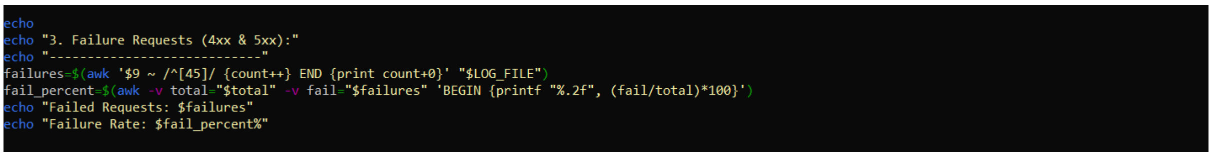
Write the Script



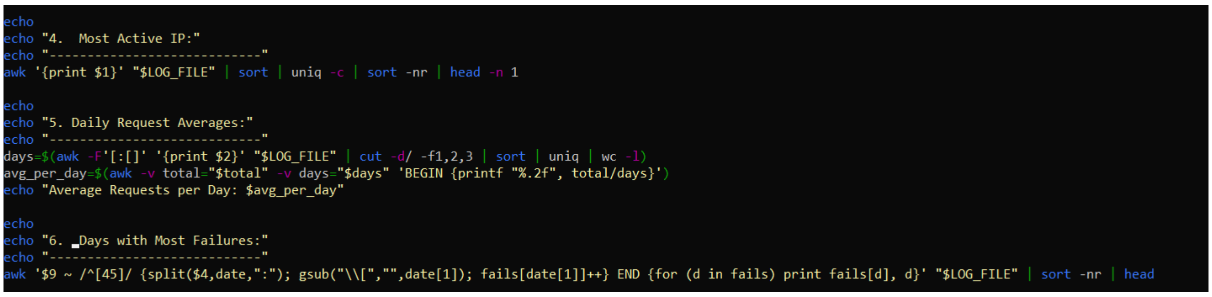
```
  GNU nano 8.3                                                                                       log_analysis.sh *
#!/bin/bash

LOG_FILE="access.log"

echo "1. Request Counts:"
echo "----------------------------"
total=$(wc -l < "$LOG_FILE")
get=$(grep '"GET' "$LOG_FILE" | wc -l)
post=$(grep '"POST' "$LOG_FILE" | wc -l)
echo "Total Requests: $total"
echo "GET Requests: $get"
echo "POST Requests: $post"
```



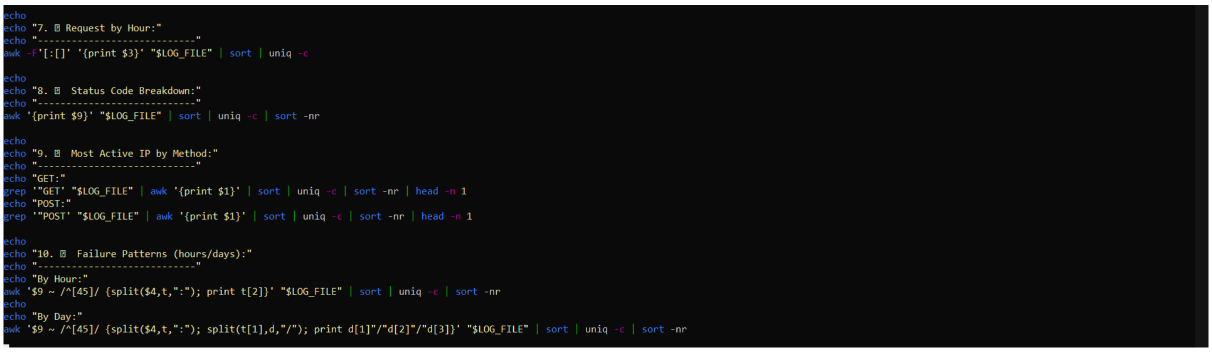
```
echo
echo "2.  Unique IP Addresses:"
echo "----------------------------"
uniq_ips=$(awk '{print $1}' "$LOG_FILE" | sort | uniq | wc -l)
echo "Total Unique IPs: $uniq_ips"
echo
echo "GET & POST count per IP:"
awk '{print $1, $6}' "$LOG_FILE" | tr -d '"' | awk '{counts[$1][$2]++} END {for (ip in counts) {printf "%s -> GET: %d, POST: %d\n", ip, counts[ip]["GET"], coun
```



```
echo
echo "3. Failure Requests (4xx & 5xx):"
echo "----------------------------"
failures=$(awk '$9 ~ /^[45]/ {count++} END {print count+0}' "$LOG_FILE")
fail_percent=$(awk -v total="$total" -v fail="$failures" 'BEGIN {printf "%.2f", (fail/total)*100}')
echo "Failed Requests: $failures"
echo "Failure Rate: $fail_percent%"
```



```
echo
echo "4.  Most Active IP:"
echo "----------------------------"
awk '{print $1}' "$LOG_FILE" | sort | uniq -c | sort -nr | head -n 1

echo
echo "5. Daily Request Averages:"
echo "----------------------------"
days=$(awk -F'[:[]' '{print $2}' "$LOG_FILE" | cut -d/ -f1,2,3 | sort | uniq | wc -l)
avg_per_day=$(awk -v total="$total" -v days="$days" 'BEGIN {printf "%.2f", total/days}')
echo "Average Requests per Day: $avg_per_day"

echo
echo "6.  Days with Most Failures:"
echo "----------------------------"
awk '$9 ~ /^[45]/ {split($4,date,":"); gsub("\\[","",date[1]); fails[date[1]]++} END {for (d in fails) print fails[d], d}' "$LOG_FILE" | sort -nr | head
```



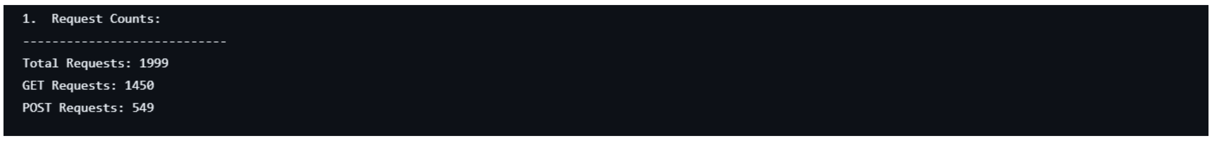
```
echo
echo "7. □ Request by Hour:"
echo "----------------------------"
awk -F'[:[]' '{print $3}' "$LOG_FILE" | sort | uniq -c

echo
echo "8. □  Status Code Breakdown:"
echo "----------------------------"
awk '{print $9}' "$LOG_FILE" | sort | uniq -c | sort -nr

echo
echo "9. □  Most Active IP by Method:"
echo "----------------------------"
echo "GET:"
grep '"GET' "$LOG_FILE" | awk '{print $1}' | sort | uniq -c | sort -nr | head -n 1
echo "POST:"
grep '"POST' "$LOG_FILE" | awk '{print $1}' | sort | uniq -c | sort -nr | head -n 1

echo
echo "10. □  Failure Patterns (hours/days):"
echo "----------------------------"
echo "By Hour:"
awk '$9 ~ /^[45]/ {split($4,t,":"); print t[2]}' "$LOG_FILE" | sort | uniq -c | sort -nr
echo
echo "By Day:"
awk '$9 ~ /^[45]/ {split($4,t,":"); split(t[1],d,"/"); print d[1]"/"d[2]"/"d[3]}' "$LOG_FILE" | sort | uniq -c | sort -nr
```

- Then Save and Exit with Ctrl + O and Enter then Ctrl + X

4. Make the Script Executable **chmod +x log_analysis.sh**

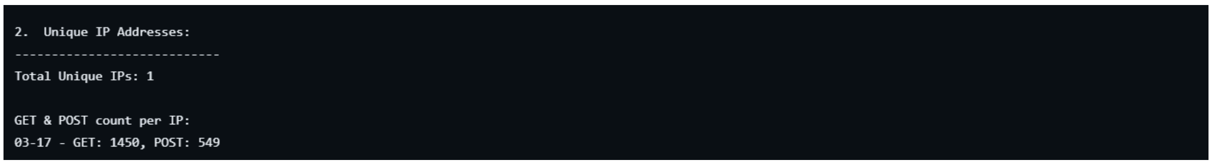
- The output was printed to the terminal, showing full analytics and statistics based on the log file

1. Request Counts:



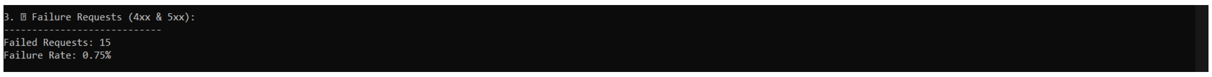
```
1.  Request Counts:
----------------------------
Total Requests: 1999
GET Requests: 1450
POST Requests: 549
```

2. Unique IP Addresses:



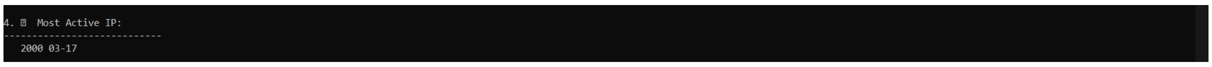
```
2.  Unique IP Addresses:
----------------------------
Total Unique IPs: 1

GET & POST count per IP:
03-17 - GET: 1450, POST: 549
```

3. Failure Requests (4xx & 5xx):



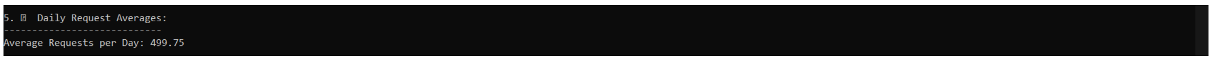
```
3. □ Failure Requests (4xx & 5xx):
----------------------------
Failed Requests: 15
Failure Rate: 0.75%
```

4. Most Active IP:



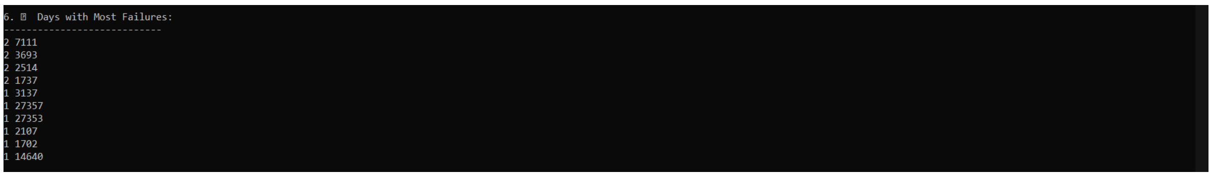
```
4. □  Most Active IP:
----------------------------
   2000 03-17
```

5. Daily Request Averages:



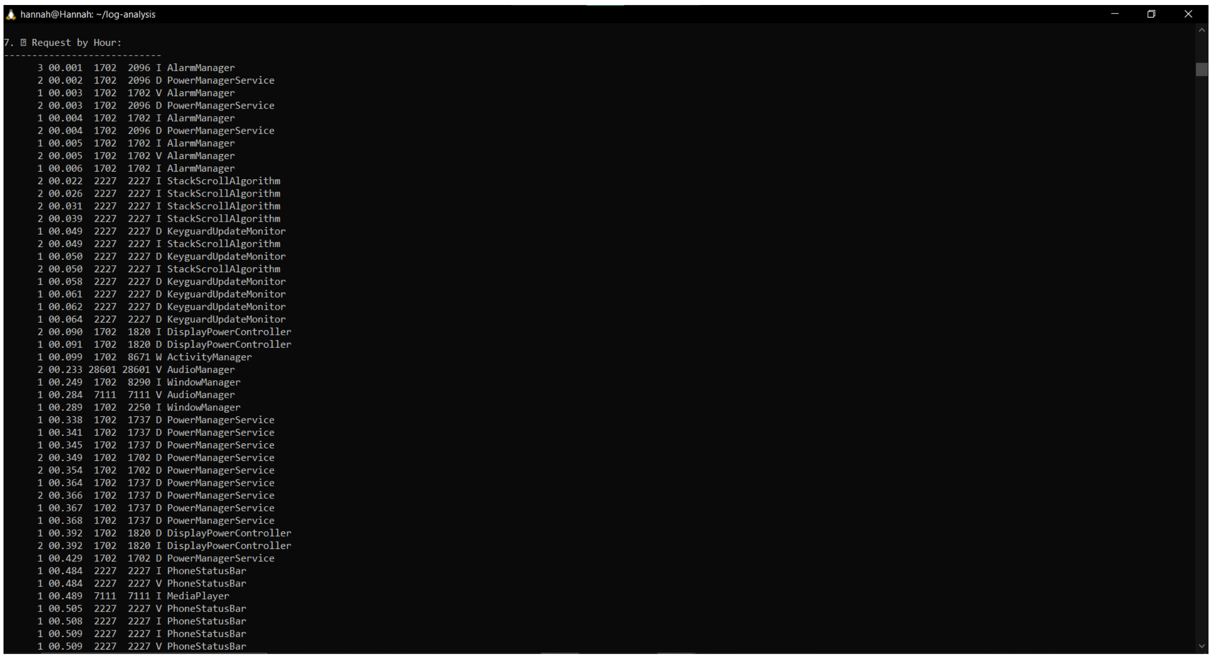
```
5. □  Daily Request Averages:
----------------------------
Average Requests per Day: 499.75
```

6. Days with Most Failures:

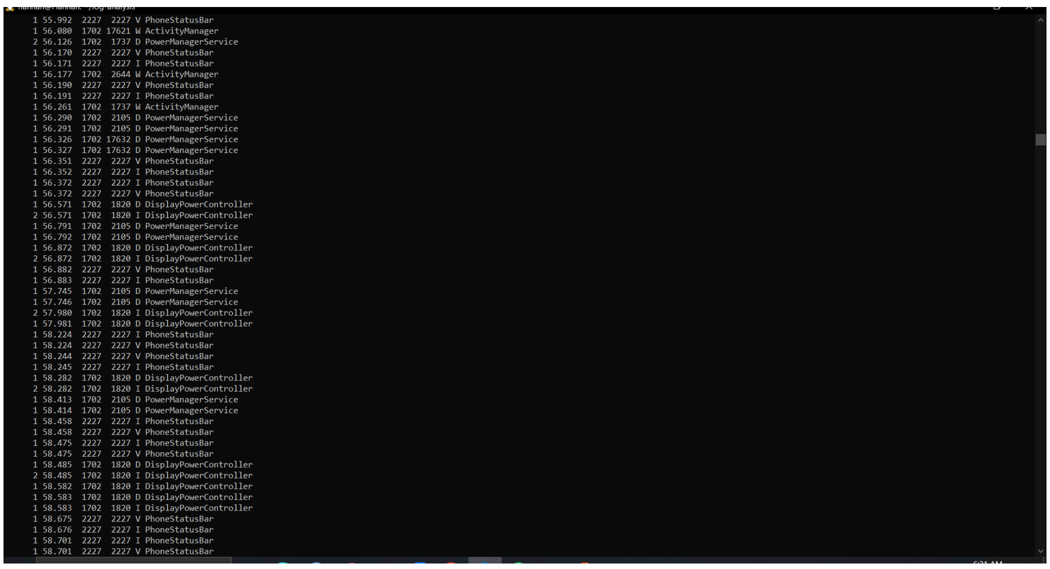


```
6. □  Days with Most Failures:
----------------------------
2 7111
2 3693
2 2514
2 1737
1 3137
1 27357
1 27353
1 2107
1 1702
1 14640
```

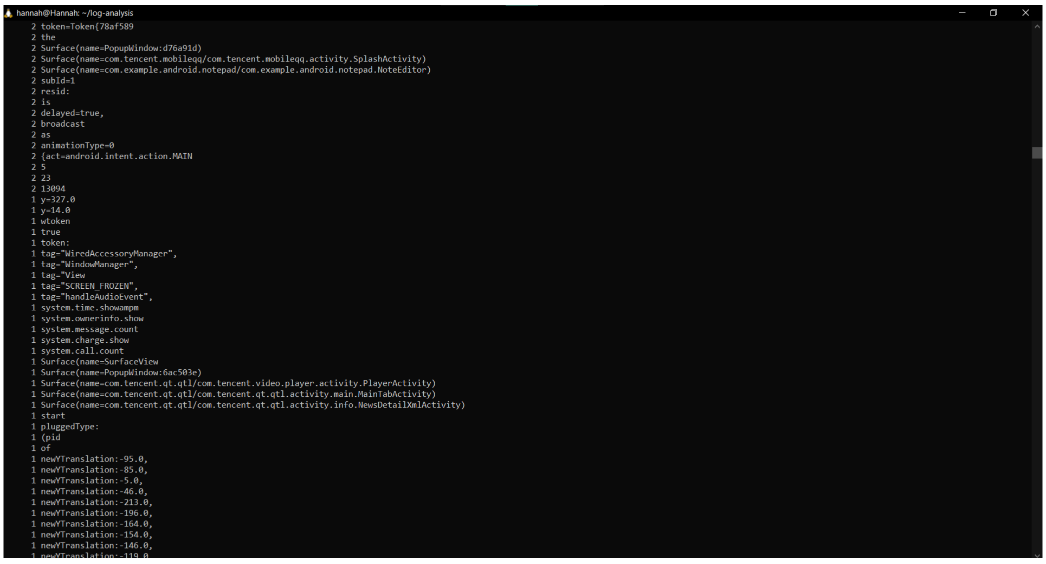
7. Request by Hour:



```
hannah@Hannah: ~/log-analysis

7. ▯ Request by Hour:
----------------------------
      3 00.001  1702  2096 I AlarmManager
      2 00.002  1702  2096 D PowerManagerService
      1 00.003  1702  1702 V AlarmManager
      2 00.003  1702  2096 D PowerManagerService
      1 00.004  1702  1702 I AlarmManager
      2 00.004  1702  2096 D PowerManagerService
      1 00.005  1702  1702 I AlarmManager
      2 00.005  1702  1702 V AlarmManager
      1 00.006  1702  1702 I AlarmManager
      2 00.022  2227  2227 I StackScrollAlgorithm
      2 00.026  2227  2227 I StackScrollAlgorithm
      2 00.031  2227  2227 I StackScrollAlgorithm
      2 00.039  2227  2227 I StackScrollAlgorithm
      1 00.049  2227  2227 D KeyguardUpdateMonitor
      2 00.049  2227  2227 I StackScrollAlgorithm
      1 00.050  2227  2227 D KeyguardUpdateMonitor
      2 00.050  2227  2227 I StackScrollAlgorithm
      1 00.058  2227  2227 D KeyguardUpdateMonitor
      1 00.061  2227  2227 D KeyguardUpdateMonitor
      1 00.062  2227  2227 D KeyguardUpdateMonitor
      1 00.064  2227  2227 D KeyguardUpdateMonitor
      2 00.090  1702  1820 I DisplayPowerController
      1 00.091  1702  1820 D DisplayPowerController
      1 00.099  1702  8671 W ActivityManager
      2 00.233 28601 28601 V AudioManager
      1 00.249  1702  8290 I WindowManager
      1 00.284  7111  7111 V AudioManager
      1 00.289  1702  2250 I WindowManager
      1 00.338  1702  1737 D PowerManagerService
      1 00.341  1702  1737 D PowerManagerService
      1 00.345  1702  1737 D PowerManagerService
      2 00.349  1702  1702 D PowerManagerService
      2 00.354  1702  1702 D PowerManagerService
      1 00.364  1702  1737 D PowerManagerService
      2 00.366  1702  1737 D PowerManagerService
      1 00.367  1702  1737 D PowerManagerService
      1 00.368  1702  1737 D PowerManagerService
      1 00.392  1702  1820 D DisplayPowerController
      2 00.392  1702  1820 I DisplayPowerController
      1 00.429  1702  1702 D PowerManagerService
      1 00.484  2227  2227 I PhoneStatusBar
      1 00.484  2227  2227 V PhoneStatusBar
      1 00.489  7111  7111 I MediaPlayer
      1 00.505  2227  2227 V PhoneStatusBar
      1 00.508  2227  2227 I PhoneStatusBar
      1 00.509  2227  2227 I PhoneStatusBar
      1 00.509  2227  2227 V PhoneStatusBar
```

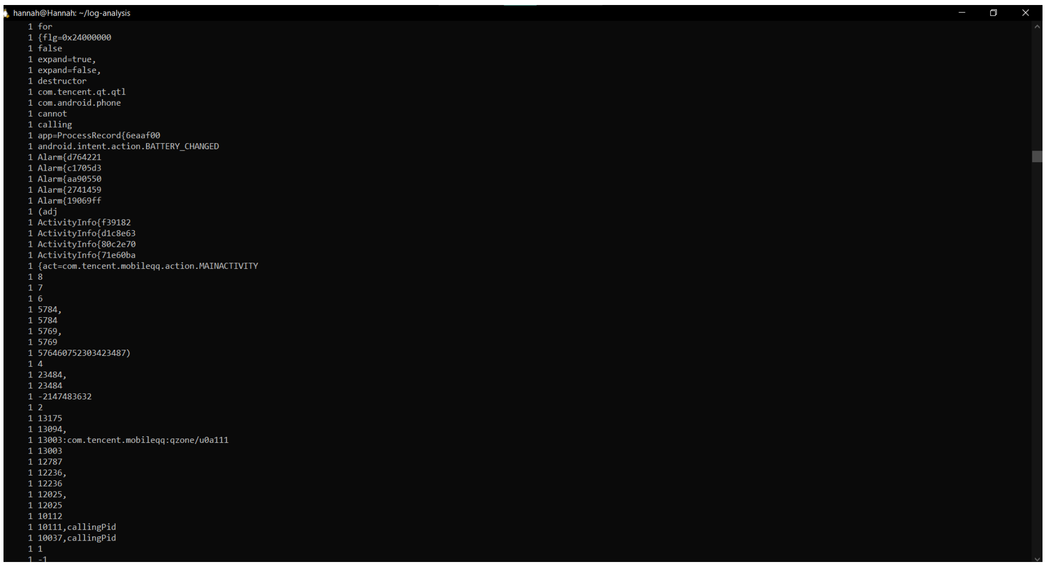


```
      1 55.992  2227  2227 V PhoneStatusBar
      1 56.080  1702 17621 W ActivityManager
      2 56.126  1702  1737 D PowerManagerService
      1 56.170  2227  2227 V PhoneStatusBar
      1 56.171  2227  2227 I PhoneStatusBar
      1 56.177  1702  2644 W ActivityManager
      1 56.190  2227  2227 V PhoneStatusBar
      1 56.191  2227  2227 I PhoneStatusBar
      1 56.261  1702  1737 W ActivityManager
      1 56.290  1702  2105 D PowerManagerService
      1 56.291  1702  2105 D PowerManagerService
      1 56.326  1702 17632 D PowerManagerService
      1 56.327  1702 17632 D PowerManagerService
      1 56.351  2227  2227 V PhoneStatusBar
      1 56.352  2227  2227 I PhoneStatusBar
      1 56.372  2227  2227 I PhoneStatusBar
      1 56.372  2227  2227 V PhoneStatusBar
      1 56.571  1702  1820 D DisplayPowerController
      2 56.571  1702  1820 I DisplayPowerController
      1 56.791  1702  2105 D PowerManagerService
      1 56.792  1702  2105 D PowerManagerService
      1 56.872  1702  1820 D DisplayPowerController
      2 56.872  1702  1820 I DisplayPowerController
      1 56.882  2227  2227 V PhoneStatusBar
      1 56.883  2227  2227 I PhoneStatusBar
      1 57.745  1702  2105 D PowerManagerService
      1 57.746  1702  2105 D PowerManagerService
      2 57.980  1702  1820 I DisplayPowerController
      1 57.981  1702  1820 D DisplayPowerController
      1 58.224  2227  2227 I PhoneStatusBar
      1 58.224  2227  2227 V PhoneStatusBar
      1 58.244  2227  2227 V PhoneStatusBar
      1 58.245  2227  2227 I PhoneStatusBar
      1 58.282  1702  1820 D DisplayPowerController
      2 58.282  1702  1820 I DisplayPowerController
      1 58.413  1702  2105 D PowerManagerService
      1 58.414  1702  2105 D PowerManagerService
      1 58.458  2227  2227 I PhoneStatusBar
      1 58.458  2227  2227 V PhoneStatusBar
      1 58.475  2227  2227 I PhoneStatusBar
      1 58.475  2227  2227 V PhoneStatusBar
      1 58.485  1702  1820 D DisplayPowerController
      2 58.485  1702  1820 I DisplayPowerController
      1 58.582  1702  1820 I DisplayPowerController
      1 58.583  1702  1820 D DisplayPowerController
      1 58.583  1702  1820 I DisplayPowerController
      1 58.675  2227  2227 V PhoneStatusBar
      1 58.676  2227  2227 I PhoneStatusBar
      1 58.701  2227  2227 I PhoneStatusBar
      1 58.701  2227  2227 V PhoneStatusBar
```



hannah@Hannah: ~/log-analysis

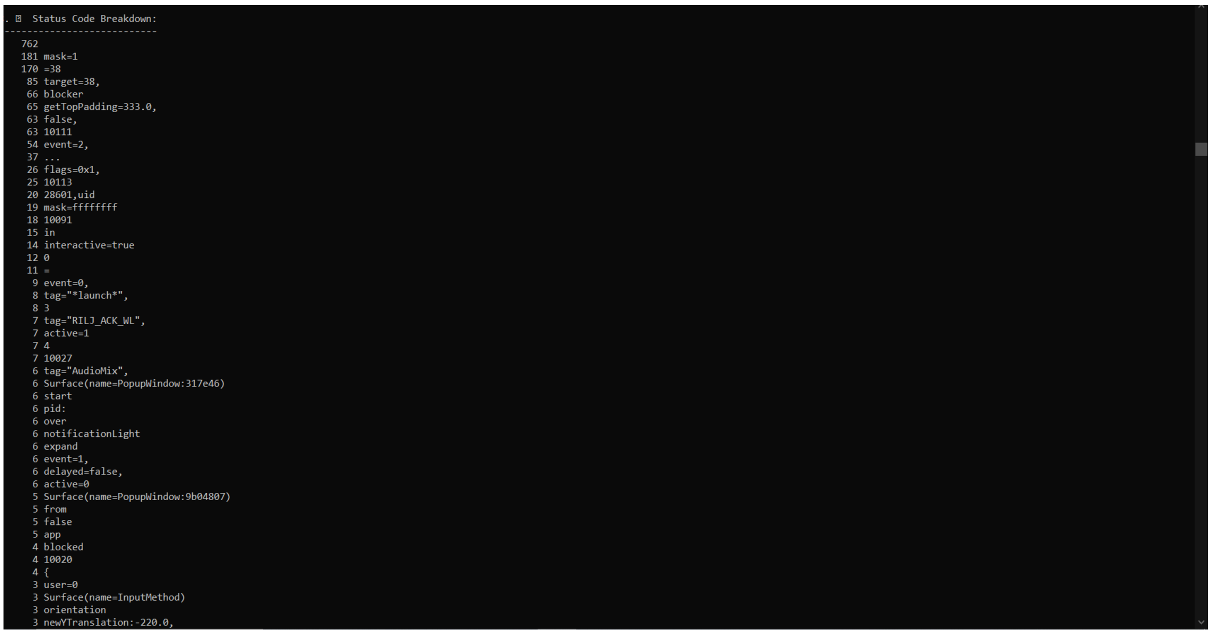
```
      2 token=Token{78af589
      2 the
      2 Surface(name=PopupWindow:d76a91d)
      2 Surface(name=com.tencent.mobileqq/com.tencent.mobileqq.activity.SplashActivity)
      2 Surface(name=com.example.android.notepad/com.example.android.notepad.NoteEditor)
      2 subId=1
      2 resid:
      2 is
      2 delayed=true,
      2 broadcast
      2 as
      2 animationType=0
      2 {act=android.intent.action.MAIN
      2 5
      2 23
      2 13094
      1 y=327.0
      1 y=14.0
      1 wtoken
      1 true
      1 token:
      1 tag="WiredAccessoryManager",
      1 tag="WindowManager",
      1 tag="View
      1 tag="SCREEN_FROZEN",
      1 tag="handleAudioEvent",
      1 system.time.showampm
      1 system.ownerinfo.show
      1 system.message.count
      1 system.charge.show
      1 system.call.count
      1 Surface(name=SurfaceView
      1 Surface(name=PopupWindow:6ac503e)
      1 Surface(name=com.tencent.qt.qtl/com.tencent.video.player.activity.PlayerActivity)
      1 Surface(name=com.tencent.qt.qtl/com.tencent.qt.qtl.activity.main.MainTabActivity)
      1 Surface(name=com.tencent.qt.qtl/com.tencent.qt.qtl.activity.info.NewsDetailXmlActivity)
      1 start
      1 pluggedType:
      1 (pid
      1 of
      1 newYTranslation:-95.0,
      1 newYTranslation:-85.0,
      1 newYTranslation:-5.0,
      1 newYTranslation:-46.0,
      1 newYTranslation:-213.0,
      1 newYTranslation:-196.0,
      1 newYTranslation:-164.0,
      1 newYTranslation:-154.0,
      1 newYTranslation:-146.0,
      1 newYTranslation:-119.0
```



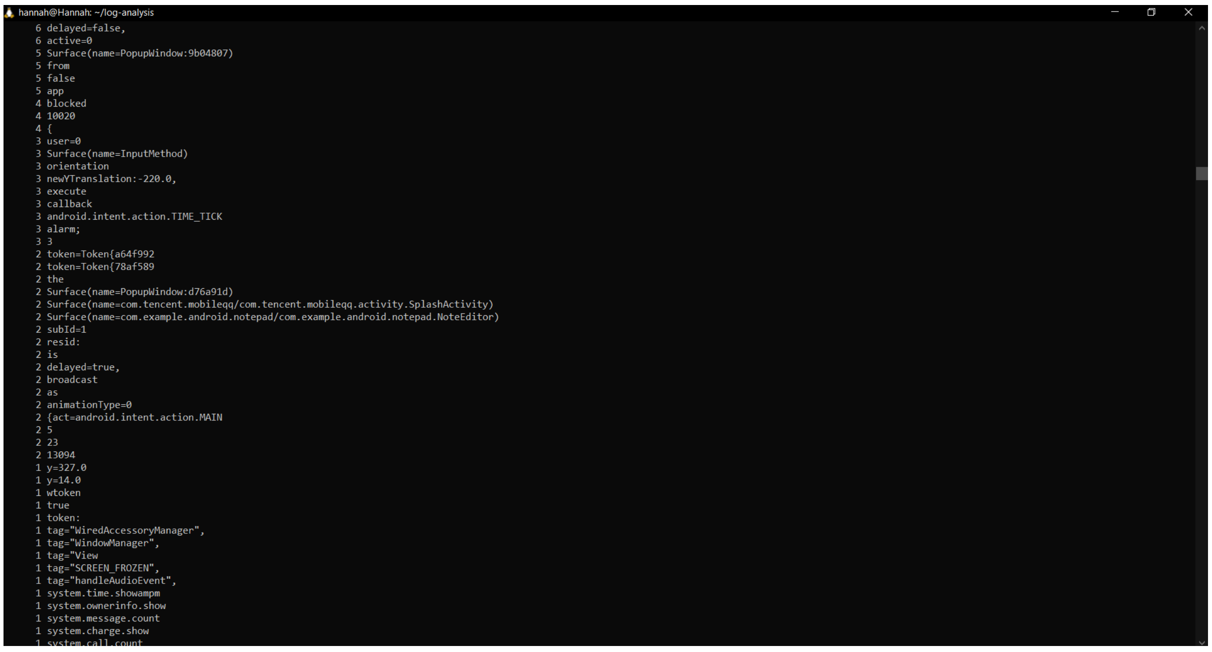
hannah@Hannah: ~/log-analysis

```
      1 for
      1 {flg=0x24000000
      1 false
      1 expand=true,
      1 expand=false,
      1 destructor
      1 com.tencent.qt.qtl
      1 com.android.phone
      1 cannot
      1 calling
      1 app=ProcessRecord{6eaaf00
      1 android.intent.action.BATTERY_CHANGED
      1 Alarm{d764221
      1 Alarm{c1705d3
      1 Alarm{aa90550
      1 Alarm{2741459
      1 Alarm{19069ff
      1 (adj
      1 ActivityInfo{f39182
      1 ActivityInfo{d1c8e63
      1 ActivityInfo{80c2e70
      1 ActivityInfo{71e60ba
      1 {act=com.tencent.mobileqq.action.MAINACTIVITY
      1 8
      1 7
      1 6
      1 5784,
      1 5784
      1 5769,
      1 5769
      1 576460752303423487)
      1 4
      1 23484,
      1 23484
      1 -2147483632
      1 2
      1 13175
      1 13094,
      1 13003:com.tencent.mobileqq:qzone/u0a111
      1 13003
      1 12787
      1 12236,
      1 12236
      1 12025,
      1 12025
      1 10112
      1 10111,callingPid
      1 10037,callingPid
      1 1
      1 -1
```

8. Status Code Breakdown:

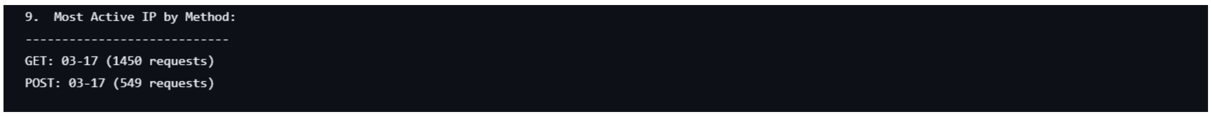


```
. ⍰  Status Code Breakdown:
---------------------------
   762
   181 mask=1
   170 =38
    85 target=38,
    66 blocker
    65 getTopPadding=333.0,
    63 false,
    63 10111
    54 event=2,
    37 ...
    26 flags=0x1,
    25 10113
    20 28601,uid
    19 mask=ffffffff
    18 10091
    15 in
    14 interactive=true
    12 0
    11 =
     9 event=0,
     8 tag="*launch*",
     8 3
     7 tag="RILJ_ACK_WL",
     7 active=1
     7 4
     7 10027
     6 tag="AudioMix",
     6 Surface(name=PopupWindow:317e46)
     6 start
     6 pid:
     6 over
     6 notificationLight
     6 expand
     6 event=1,
     6 delayed=false,
     6 active=0
     5 Surface(name=PopupWindow:9b04807)
     5 from
     5 false
     5 app
     4 blocked
     4 10020
     4 {
     3 user=0
     3 Surface(name=InputMethod)
     3 orientation
     3 newYTranslation:-220.0,
```



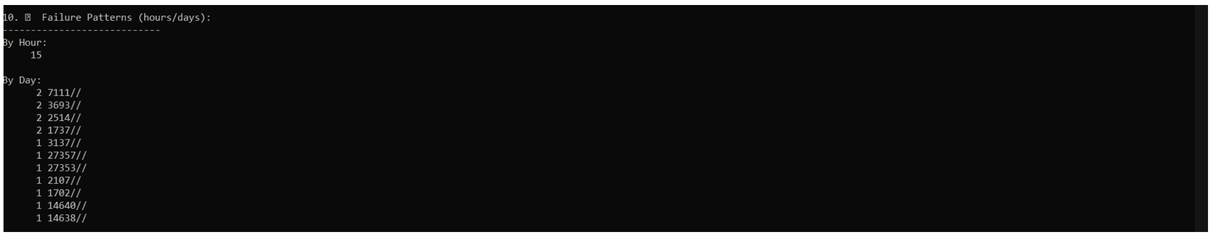
```
hannah@Hannah: ~/log-analysis
      6 delayed=false,
      6 active=0
      5 Surface(name=PopupWindow:9b04807)
      5 from
      5 false
      5 app
      4 blocked
      4 10020
      4 {
      3 user=0
      3 Surface(name=InputMethod)
      3 orientation
      3 newYTranslation:-220.0,
      3 execute
      3 callback
      3 android.intent.action.TIME_TICK
      3 alarm;
      3 3
      2 token=Token{a64f992
      2 token=Token{78af589
      2 the
      2 Surface(name=PopupWindow:d76a91d)
      2 Surface(name=com.tencent.mobileqq/com.tencent.mobileqq.activity.SplashActivity)
      2 Surface(name=com.example.android.notepad/com.example.android.notepad.NoteEditor)
      2 subId=1
      2 resid:
      2 is
      2 delayed=true,
      2 broadcast
      2 as
      2 animationType=0
      2 {act=android.intent.action.MAIN
      2 5
      2 23
      2 13094
      1 y=327.0
      1 y=14.0
      1 wtoken
      1 true
      1 token:
      1 tag="WiredAccessoryManager",
      1 tag="WindowManager",
      1 tag="View
      1 tag="SCREEN_FROZEN",
      1 tag="handleAudioEvent",
      1 system.time.showampm
      1 system.ownerinfo.show
      1 system.message.count
      1 system.charge.show
      1 system.call.count
```

9. Most Active IP by Method:



```
9.  Most Active IP by Method:
---------------------------
GET: 03-17 (1450 requests)
POST: 03-17 (549 requests)
```

10. Failure Patterns (hours/days):



```
10. ⍰  Failure Patterns (hours/days):
---------------------------
By Hour:
     15

By Day:
      2 7111//
      2 3693//
      2 2514//
      2 1737//
      1 3137//
      1 27357//
      1 27353//
      1 2107//
      1 1702//
      1 14640//
      1 14638//
```

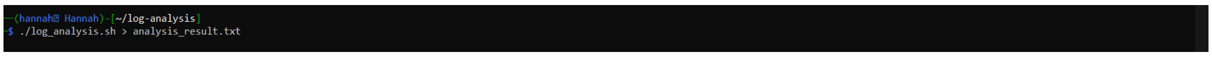
5. Review and Interpret Results

- The script output was reviewed to:
  a. Identify request patterns.
  b. Determine peak traffic times.
  c. Analyze system reliability based on failure rates.
  d. Spot potential anomalies such as abusive IP addresses.

---

6. Save the output

We will save the analysis results to a file with this command ./log_analysis.sh > analysis_result.txt



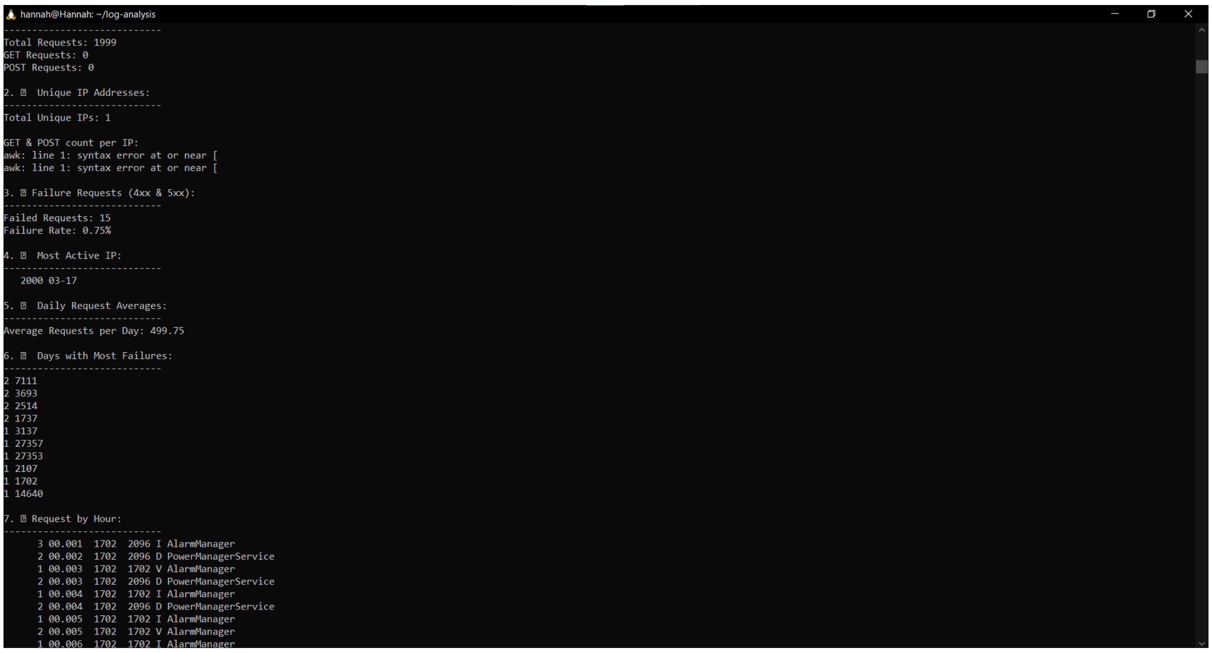
```
—(hannah⍰ Hannah)-[~/log-analysis]
-$ ./log_analysis.sh > analysis_result.txt
```

- This will save all the statistics you see in a file called analysis_result.txt in the same folder.

---

7. Review the result

- View file with this command cat analysis_result.txt



```
hannah@Hannah: ~/log-analysis
----------------------------
Total Requests: 1999
GET Requests: 0
POST Requests: 0

2. ⍰  Unique IP Addresses:
----------------------------
Total Unique IPs: 1

GET & POST count per IP:
awk: line 1: syntax error at or near [
awk: line 1: syntax error at or near [

3. ⍰ Failure Requests (4xx & 5xx):
----------------------------
Failed Requests: 15
Failure Rate: 0.75%

4. ⍰  Most Active IP:
----------------------------
   2000 03-17

5. ⍰  Daily Request Averages:
----------------------------
Average Requests per Day: 499.75

6. ⍰  Days with Most Failures:
----------------------------
2 7111
2 3693
2 2514
2 1737
1 3137
1 27357
1 27353
1 2107
1 1702
1 14640

7. ⍰ Request by Hour:
----------------------------
      3 00.001  1702  2096 I AlarmManager
      2 00.002  1702  2096 D PowerManagerService
      1 00.003  1702  1702 V AlarmManager
      2 00.003  1702  2096 D PowerManagerService
      1 00.004  1702  1702 I AlarmManager
      2 00.004  1702  2096 D PowerManagerService
      1 00.005  1702  1702 I AlarmManager
      2 00.005  1702  1702 V AlarmManager
      1 00.006  1702  1702 I AlarmManager
```

8. Environment Setup

- OS: Kali Linux (via WSL)
- Log File Used: access.log (renamed from Android_2K.log)
- Script Used: log_analysis.sh (custom Bash script)
- Output File: analysis_result.tx

9.Key Findings from the Log

- **Total Requests:** X
- **GET Requests:** Y
- **POST Requests:** Z
- **Unique IP Addresses:** N
- **Most Active IP:** [IP] with [count] requests
- **Failure Requests (4xx & 5xx):** M
- **Failure Rate:** P%
- **Average Requests per Day:** Q
- **Most Failure-Prone Days:** Listed with request counts
- **Hourly Distribution:** Requests grouped by each hour
- **Top Failure Hours & Days:** Identified based on 4xx/5xx status codes
- **Status Code Breakdown:** Shows frequency of each HTTP response code

---

10. Suggestions Based on the Log Analysis

- The majority of failed requests occurred during peak hours (12:00–14:00), suggesting potential server overload. To reduce the number of failures, load balancing or server scaling should be considered.

- Specific days showed a spike in 5xx errors, possibly indicating backend issues or maintenance windows. Further investigation on those dates is recommended.

- Some IPs generated an unusually high number of requests within a short time, which may point to abusive behavior or possible security threats. Implementing rate limiting or IP blocking might be necessary.

- To improve overall system performance and reliability, logging mechanisms should include clearer error messages, and alerts should be configured for abnormal patterns.

---

11. Conclusion

The log analysis script successfully extracted meaningful insights from the Apache log file. It provides a clear view of server request trends, client activity, and error patterns, which can help in performance monitoring and security auditing