INFORMATION SECURITY MANAGMENT

NAME: MERAL AYMAN ABBAS
ID: 2205116

- WHAT CAN BE DONE TO REDUCE THE NUMBER OF FAILURES?

IMPROVE CODE QUALITY – TEST THOROUGHLY, FIX BUGS, AND USE ERROR HANDLING.

MONITOR & ANALYZE LOGS – DETECT ISSUES EARLY WITH ALERTS AND LOG TOOLS.

MAINTAIN SYSTEMS – UPDATE SOFTWARE, OPTIMIZE RESOURCES, AND CHECK DEPENDENCIES.

AUTOMATE RECOVERY – USE ROLLBACKS, AND FAILOVER MECHANISMS.

FIX 404 ERRORS: UPDATE WEBSITE TO FIX DEAD LINKS OR GUIDE USERS TO CORRECT PAGES

.**FIX 403 ERRORS:** DOUBLE-CHECK ACCESS RULES; SHOW FRIENDLY ERROR MESSAGES

FIX 500 ERRORS: DEBUG APP CODE, CHECK SERVER HEALTH, OR BOOST MEMORY/CPU.

• **QUICK FIX:** ADD ALERTS FOR ERRORS AND TEST APP UPDATES BEFORE DEPLOYMENT.

WHICH DAYS OR TIMES NEED ATTENTION BASED ON THE REQUEST PATTERNS AND FAILURE TRENDS?

HIGH-FAILURE DAYS – CHECK LOGS FOR DATES WITH SPIKES (E.G., 10/MAY/2025). INVESTIGATE CAUSES LIKE UPDATES, ATTACKS, OR OUTAGES.

PEAK TRAFFIC HOURS – MONITOR "REQUESTS BY HOUR" (E.G., 14:00–16:00). SERVER STRESS MAY INCREASE FAILURES DURING THESE TIMES.

WEEKLY TRENDS – WEEKDAYS (ESPECIALLY MONDAYS) OFTEN SEE HIGHER LOADS THAN WEEKENDS.

POST-UPDATE PERIODS – ERRORS MAY RISE AFTER DEPLOYMENTS (CHECK LOGS POST-RELEASE).

ACTIONS:

SCALE RESOURCES DURING PEAK HOURS.

FIX RECURRING ERRORS ON HIGH-FAILURE DAYS.

SCHEDULE MAINTENANCE/UPDATES DURING LOW-TRAFFIC PERIODS (E.G., LATE NIGHTS/WEEKENDS).

SET UP AUTOMATED ALERTS FOR ANOMALY DETECTION.

- ANY SECURITY CONCERNS OR ANOMALIES (E.G., MULTIPLE REQUESTS FROM THE SAME IP WITHIN A SHORT TIME SPAN).

SUSPICIOUS IPS – CHECK "TOP USERS" FOR ABNORMAL ACTIVITY

REQUESTS IN AN HOUR—LIKELY A BOT OR BRUTE-FORCE ATTACK).

UNUSUAL REQUEST PATTERNS – WATCH FOR:

- EXCESSIVE POST/PUT/DELETE REQUESTS (COULD INDICATE DATA SCRAPING OR INJECTION ATTEMPTS).
- HIGH 403/404/5XX ERRORS FROM A SINGLE IP (MAY SIGNAL PROBING FOR VULNERABILITIES).
- RAPID SEQUENTIAL LOGINS (POSSIBLE CREDENTIAL STUFFING).
-

GEOGRAPHIC ANOMALIES – SUDDEN TRAFFIC FROM UNEXPECTED COUNTRIES (E.G., A USER'S ACCOUNT ACCESSED FROM MULTIPLE REGIONS IN MINUTES).

ODD TIMESTAMPS – MIDNIGHT OR LOW-TRAFFIC-HOUR SPIKES (COMMON FOR AUTOMATED ATTACKS).

ACTIONS:

RATE LIMITING (E.G., BLOCK IPS EXCEEDING 500 REQUESTS/MINUTE).

AUTOMATED BANS FOR REPEATED FAILED LOGINS (E.G., 5+ 403S IN 10 MINS).

BOT MITIGATION (RECAPTCHA, CLOUDFLARE, OR FINGERPRINTING).

REVIEW SERVER LOGS FOR SQLI/XSS PAYLOADS IN URLS.

SUGGESTIONS FOR IMPROVING THE SYSTEM OR SERVICE BASED ON THE ANALYSIS.

FASTER PERFORMANCE – CACHE CONTENT, OPTIMIZE CODE.
MORE STABLE – AUTO-SCALE SERVERS, LOG ERRORS BETTER, PREP BACKUPS.
STRONGER SECURITY – BLOCK SUSPICIOUS IPS, RATE-LIMIT LOGINS, ENCRYPT DATA.
BETTER UX – HELPFUL ERROR PAGES, AUTO-RETRY FAILED REQUESTS.
SMARTER UPDATES – DEPLOY DURING LOW-TRAFFIC HOURS, TEST GRADUALLY.
PROACTIVE MONITORING – SET UP REAL-TIME ALERTS FOR UNUSUAL TRAFFIC OR ERRORS.
REGULAR AUDITS – REVIEW LOGS WEEKLY TO CATCH RECURRING ISSUES EARLY.

CREATE A FILE TO INCLUDE THE BASH COMAND IN IT:

touch analyze_log.sh
nano analyze_log.sh

wc -l acc.log

TO TELL HOW MANY ENTITES IN THE LOGFILE I USED

THIS IS MY BASH FILE:

```bash
#!/bin/bash

LOG_FILE="acc.log"
REPORT_FILE="log_analysis_report.txt"


echo "Log File Analysis Report" > $REPORT_FILE
echo "Generated on: $(date)" >> $REPORT_FILE
echo "==================================" >> $REPORT_FILE

# 1. Request Counts
echo -e "\n1. Request Counts" >> $REPORT_FILE
TOTAL_REQUESTS=$(wc -l < $LOG_FILE)
GET_REQUESTS=$(grep '"GET' $LOG_FILE | wc -l)
POST_REQUESTS=$(grep '"POST' $LOG_FILE | wc -l)
echo "Total Requests: $TOTAL_REQUESTS" >> $REPORT_FILE
echo "GET Requests: $GET_REQUESTS" >> $REPORT_FILE
echo "POST Requests: $POST_REQUESTS" >> $REPORT_FILE

# 2. Unique IP Addresses
echo -e "\n2. Unique IP Addresses" >> $REPORT_FILE
UNIQUE_IPS=$(awk '{print $1}' $LOG_FILE | sort | uniq)
UNIQUE_IP_COUNT=$(echo "$UNIQUE_IPS" | wc -l)
echo "Total Unique IPs: $UNIQUE_IP_COUNT" >> $REPORT_FILE
echo "IP Address | GET Requests | POST Requests" >> $REPORT_FILE
echo "-------------------------------------" >> $REPORT_FILE
for IP in $UNIQUE_IPS; do
    IP_GET=$(grep $IP $LOG_FILE | grep '"GET' | wc -l)
    IP_POST=$(grep $IP $LOG_FILE | grep '"POST' | wc -l)
    echo "$IP | $IP_GET | $IP_POST" >> $REPORT_FILE
done

# 3. Failure Requests
echo -e "\n3. Failure Requests" >> $REPORT_FILE
FAILED_REQUESTS=$(awk '$9 ~ /^[45]/' $LOG_FILE | wc -l)
FAILED_PERCENT=$(echo "scale=2; ($FAILED_REQUESTS / $TOTAL_REQUESTS) * 100" | bc)
echo "Failed Requests (4xx/5xx): $FAILED_REQUESTS" >> $REPORT_FILE
echo "Percentage of Failed Requests: $FAILED_PERCENT%" >> $REPORT_FILE

# 4. Top User
echo -e "\n4. Top User" >> $REPORT_FILE
TOP_IP=$(awk '{print $1}' $LOG_FILE | sort | uniq -c | sort -nr | head -1 | awk '{print $2}')
TOP_IP_COUNT=$(awk '{print $1}' $LOG_FILE | sort | uniq -c | sort -nr | head -1 | awk '{print $1}')
echo "Most Active IP: $TOP_IP with $TOP_IP_COUNT requests" >> $REPORT_FILE

# 5. Daily Request Averages
echo -e "\n5. Daily Request Averages" >> $REPORT_FILE
DAYS=$(awk -F'[' '{print $2}' $LOG_FILE | awk -F: '{print $1}' | sort | uniq)
DAY_COUNT=$(echo "$DAYS" | wc -l)
AVG_REQUESTS=$(echo "scale=2; $TOTAL_REQUESTS / $DAY_COUNT" | bc)
echo "Average Requests per Day: $AVG_REQUESTS" >> $REPORT_FILE

# 6. Failure Analysis
echo -e "\n6. Failure Analysis" >> $REPORT_FILE
echo "Days with Highest Failure Requests:" >> $REPORT_FILE
awk '$9 ~ /^[45]/ {print $0}' $LOG_FILE | awk -F'[' '{print $2}' | awk -F: '{print $1}' | sort | uniq -c | sort -nr | head -3 >> $REPORT_FILE

# 7. Request by Hour
echo -e "\n7. Request by Hour" >> $REPORT_FILE
echo "Hour | Requests" >> $REPORT_FILE
echo "---------------" >> $REPORT_FILE
awk -F: '{print $2}' $LOG_FILE | awk '{print $1}' | sort | uniq -c | awk '{printf "%02d | %d\n", $2, $1}' >> $REPORT_FILE

# 8. Status Codes Breakdown
echo -e "\n8. Status Codes Breakdown" >> $REPORT_FILE
echo "Status Code | Count" >> $REPORT_FILE
echo "-------------------" >> $REPORT_FILE
awk '{print $9}' $LOG_FILE | sort | uniq -c | awk '{print $2 " | " $1}' >> $REPORT_FILE

# 9. Most Active User by Method
echo -e "\n9. Most Active User by Method" >> $REPORT_FILE
TOP_GET_IP=$(awk '/"GET/ {print $1}' $LOG_FILE | sort | uniq -c | sort -nr | head -1 | awk '{print $2}')
TOP_GET_COUNT=$(awk '/"GET/ {print $1}' $LOG_FILE | sort | uniq -c | sort -nr | head -1 | awk '{print $1}')
TOP_POST_IP=$(awk '/"POST/ {print $1}' $LOG_FILE | sort | uniq -c | sort -nr | head -1 | awk '{print $2}')
TOP_POST_COUNT=$(awk '/"POST/ {print $1}' $LOG_FILE | sort | uniq -c | sort -nr | head -1 | awk '{print $1}')
echo "Most Active GET IP: $TOP_GET_IP with $TOP_GET_COUNT requests" >> $REPORT_FILE
echo "Most Active POST IP: $TOP_POST_IP with $TOP_POST_COUNT requests" >> $REPORT_FILE

# 10. Patterns in Failure Requests
echo -e "\n10. Patterns in Failure Requests" >> $REPORT_FILE
echo "Hours with Highest Failure Requests:" >> $REPORT_FILE
awk '$9 ~ /^[45]/ {print $0}' $LOG_FILE | awk -F: '{print $2}' | awk '{print $1}' | sort | uniq -c | sort -nr | head -3 >> $REPORT_FILE

# 11. Analysis Suggestions
echo -e "\n11. Analysis Suggestions" >> $REPORT_FILE
echo "- Investigate IPs with high failure rates for potential malicious activity." >> $REPORT_FILE
echo "- Schedule maintenance during low-traffic hours (based on hourly request trends)." >> $REPORT_FILE
echo "- Monitor days with high failures for server or application issues." >> $REPORT_FILE
echo "- Implement rate-limiting for IPs with unusually high request counts." >> $REPORT_FILE

# pandoc $REPORT_FILE -o log_analysis_report.pdf

echo "Analysis complete. Report saved to $REPORT_FILE"
```

1. **REQUEST COUNT:**

```
ECHO -E "\N1. REQUEST COUNTS" >> $REPORT_FILE
TOTAL_REQUESTS=$(WC -L < $LOG_FILE)
GET_REQUESTS=$(GREP '"GET' $LOG_FILE | WC -L)
POST_REQUESTS=$(GREP '"POST' $LOG_FILE | WC -L)
```

SAMPLE RUN:

TOTAL REQUESTS: 20975
GET REQUESTS: 15939
POST REQUESTS: 4989

PURPOSE: COUNTS TOTAL, GET, AND POST REQUESTS TO UNDERSTAND THE VOLUME AND TYPE OF TRAFFIC.

COMMANDS:

WC -L < "$LOG_FILE": COUNTS TOTAL LINES (REQUESTS) IN THE LOG FILE.
GREP "GET" "$LOG_FILE" | WC -L: COUNTS LINES CONTAINING "GET" FOR GET REQUESTS.
GREP "POST" "$LOG_FILE" | WC -L: COUNTS LINES CONTAINING "POST" FOR POST REQUESTS.

2. UNIQUE IP ADDRESSES:

```
ECHO -E "\N2. UNIQUE IP ADDRESSES" >> $REPORT_FILE
UNIQUE_IPS=$(AWK '{PRINT $1}' $LOG_FILE | SORT | UNIQ)
UNIQUE_IP_COUNT=$(ECHO "$UNIQUE_IPS" | WC -L)
ECHO "TOTAL UNIQUE IPS: $UNIQUE_IP_COUNT" >> $REPORT_FILE
ECHO "IP ADDRESS | GET REQUESTS | POST REQUESTS" >> $REPORT_FILE
ECHO "----------------------------------------" >> $REPORT_FILE
FOR IP IN $UNIQUE_IPS; DO
IP_GET=$(GREP $IP $LOG_FILE | GREP '"GET' | WC -L)
IP_POST=$(GREP $IP $LOG_FILE | GREP '"POST' | WC -L)
ECHO "$IP | $IP_GET | $IP_POST" >> $REPORT_FILE
DONE
```

SAMPLE RUN:

TOTAL UNIQUE IPS: 1807
IP ADDRESS | GET REQUESTS | POST REQUESTS

10.0.0.1 | 715 | 697
100.2.4.116 | 6 | 0
100.43.83.137 | 89 | 0
101.119.18.35 | 33 | 0
101.199.108.50 | 3 | 0
101.226.168.196 | 1 | 0
101.226.168.198 | 1 | 0
101.226.33.222 | 2 | 0
103.245.44.13 | 4 | 0

PURPOSE: IDENTIFIES THE NUMBER OF UNIQUE CLIENTS ACCESSING THE SERVER.
COMMAND: AWK '{PRINT $1}' "$LOG_FILE" | SORT | UNIQ | WC -L EXTRACTS THE FIRST FIELD (IP), SORTS, REMOVES DUPLICATES, AND COUNTS UNIQUE IPS

3. FAILURE REQUESTS

```
ECHO -E "\N3. FAILURE REQUESTS" >> $REPORT_FILE
FAILED_REQUESTS=$(AWK '$9 ~ /^[45]/' $LOG_FILE | WC -L)
FAILED_PERCENT=$(ECHO "SCALE=2; ($FAILED_REQUESTS / $TOTAL_REQUESTS) * 100" | BC)
ECHO "FAILED REQUESTS (4XX/5XX): $FAILED_REQUESTS" >> $REPORT_FILE
ECHO "PERCENTAGE OF FAILED REQUESTS: $FAILED_PERCENT%" >> $REPORT_FILE
```

SAMPLE RUN:

```
FAILED REQUESTS (4XX/5XX): 7197
PERCENTAGE OF FAILED REQUESTS: 34.00%
```

PURPOSE: QUANTIFIES FAILED REQUESTS (4XX/5XX STATUS CODES) AND THEIR PERCENTAGE OF TOTAL REQUESTS.

COMMANDS:

AWK '$9 ~ /^[45][0-9][0-9]$/' "$LOG_FILE" | WC -L: COUNTS REQUESTS WITH STATUS CODES STARTING WITH 4 OR 5.

ECHO "SCALE=2; ($FAILED_REQUESTS / $TOTAL_REQUESTS) * 100" | BC: CALCULATES THE FAILURE PERCENTAGE WITH TWO DECIMAL PLACES.

4. TOP USER:

```
ECHO -E "\N4. TOP USER" >> $REPORT_FILE
TOP_IP=$(AWK '{PRINT $1}' $LOG_FILE | SORT | UNIQ -C | SORT -NR | HEAD -1 | AWK '{PRINT $2}')
TOP_IP_COUNT=$(AWK '{PRINT $1}' $LOG_FILE | SORT | UNIQ -C | SORT -NR | HEAD -1 | AWK '{PRINT $1}')
ECHO "MOST ACTIVE IP: $TOP_IP WITH $TOP_IP_COUNT REQUESTS" >> $REPORT_FILE
```

SAMPLE RUN:

MOST ACTIVE IP: 1.1.1.1 WITH 1470 REQUESTS

PURPOSE: FINDS THE IP WITH THE MOST REQUESTS, INDICATING POTENTIAL HEAVY USERS OR BOTS.
COMMAND: AWK '{PRINT $1}' "$LOG_FILE" | SORT | UNIQ -C | SORT -NR | HEAD -1 COUNTS IPS, SORTS BY COUNT (DESCENDING), AND TAKES THE TOP ONE.

5. DAILY REQUEST AVERAGES

```
ECHO -E "\N5. DAILY REQUEST AVERAGES" >> $REPORT_FILE
DAYS=$(AWK -F'[' '{PRINT $2}' $LOG_FILE | AWK -F: '{PRINT $1}' | SORT | UNIQ)
DAY_COUNT=$(ECHO "$DAYS" | WC -L)
AVG_REQUESTS=$(ECHO "SCALE=2; $TOTAL_REQUESTS / $DAY_COUNT" | BC)
ECHO "AVERAGE REQUESTS PER DAY: $AVG_REQUESTS" >> $REPORT_FILE
```

SAMPLE RUN:

AVERAGE REQUESTS PER DAY: 2330.55

PURPOSE: CALCULATES AVERAGE REQUESTS PER DAY TO UNDERSTAND DAILY TRAFFIC PATTERNS.
COMMANDS:
AWK -F'[' '{PRINT $2}' "$LOG_FILE" | AWK -F: '{PRINT $1}' | SORT | UNIQ | WC -L: COUNTS UNIQUE DAYS.
ECHO "SCALE=2; $TOTAL_REQUESTS / $DAYS" | BC: COMPUTES AVERAGE REQUESTS PER DAY.

6. FAILURE ANALYSIS

```
ECHO -E "\N6. FAILURE ANALYSIS" >> $REPORT_FILE
ECHO "DAYS WITH HIGHEST FAILURE REQUESTS:" >> $REPORT_FILE
AWK '$9 ~ /^[45]/ {PRINT $0}' $LOG_FILE | AWK -F'[' '{PRINT $2}' | AWK -F: '{PRINT $1}' |
SORT | UNIQ -C | SORT -NR | HEAD -3 >> $REPORT_FILE
```

SAMPLE RUN:

DAYS WITH HIGHEST FAILURE REQUESTS:
1786 10/MAY/2025
1746 11/MAY/2025
1731 09/MAY/2025

PURPOSE: IDENTIFIES DAYS WITH THE MOST FAILED REQUESTS FOR TARGETED INVESTIGATION.
COMMAND: AWK '$9 ~ /^[45][0-9][0-9]$/ {PRINT $4}' "$LOG_FILE" | SORT | UNIQ -C | SORT -NR | HEAD -3 LISTS TOP THREE DAYS WITH FAILURES.

7. REQUEST BY HOUR:

```
ECHO -E "\N7. REQUEST BY HOUR" >> $REPORT_FILE
ECHO "HOUR | REQUESTS" >> $REPORT_FILE
ECHO "----------------" >> $REPORT_FILE
AWK -F: '{PRINT $2}' $LOG_FILE | AWK '{PRINT $1}' | SORT | UNIQ -C | AWK '{PRINTF "%02D | %D\N", $2, $1}' >>
$REPORT_FILE
```

SAMPLE RUN:

```
HOUR | REQUESTS
----------------
00 | 777
01 | 777
02 | 786
03 | 1114
04 | 785
05 | 802
06 | 806
07 | 711
08 | 766
```

PURPOSE: SHOWS REQUEST DISTRIBUTION BY HOUR TO IDENTIFY PEAK TIMES.
COMMAND: AWK -F'[:[]' '{PRINT $3}' "$LOG_FILE" | SORT | UNIQ -C | AWK '{PRINTF "%02D | %D\N", $2, $1}'
EXTRACTS HOURS, COUNTS REQUESTS, AND FORMATS OUTPUT.

8. STATUS CODES BREAKDOWN

```
ECHO -E "\N8. STATUS CODES BREAKDOWN" >> $REPORT_FILE
ECHO "STATUS CODE | COUNT" >> $REPORT_FILE
ECHO "--------------------" >> $REPORT_FILE
AWK '{PRINT $9}' $LOG_FILE | SORT | UNIQ -C | AWK '{PRINT $2 " | " $1}' >> $REPORT_FILE
```

SAMPLE RUN:

```
STATUS CODE | COUNT
--------------------
200 | 10991
201 | 1000
206 | 62
301 | 1263
302 | 7
304 | 455
400 | 994
```

PURPOSE: SUMMARIZES THE FREQUENCY OF EACH HTTP STATUS CODE.
COMMAND: AWK '{PRINT $9}' "$LOG_FILE" | SORT | UNIQ -C | AWK '{PRINT $2 " | " $1}' COUNTS OCCURRENCES OF EACH STATUS CODE.

9. MOST ACTIVE USER BY METHOD:

```
ECHO -E "\N9. MOST ACTIVE USER BY METHOD" >> $REPORT_FILE
TOP_GET_IP=$(AWK '/"GET/ {PRINT $1}' $LOG_FILE | SORT | UNIQ -C | SORT -NR | HEAD -1 | AWK '{PRINT $2}')
TOP_GET_COUNT=$(AWK '/"GET/ {PRINT $1}' $LOG_FILE | SORT | UNIQ -C | SORT -NR | HEAD -1 | AWK '{PRINT $1}')
TOP_POST_IP=$(AWK '/"POST/ {PRINT $1}' $LOG_FILE | SORT | UNIQ -C | SORT -NR | HEAD -1 | AWK '{PRINT $2}')
TOP_POST_COUNT=$(AWK '/"POST/ {PRINT $1}' $LOG_FILE | SORT | UNIQ -C | SORT -NR | HEAD -1 | AWK '{PRINT $1}')
ECHO "MOST ACTIVE GET IP: $TOP_GET_IP WITH $TOP_GET_COUNT REQUESTS" >> $REPORT_FILE
ECHO "MOST ACTIVE POST IP: $TOP_POST_IP WITH $TOP_POST_COUNT REQUESTS" >> $REPORT_FILE
```

SAMPLE RUN:

```
MOST ACTIVE GET IP: 192.168.1.1 WITH 731 REQUESTS
MOST ACTIVE POST IP: 1.1.1.1 WITH 750 REQUESTS
```

PURPOSE: IDENTIFIES THE MOST ACTIVE IPS FOR GET AND POST REQUESTS SEPARATELY.

COMMANDS:

AWK '/GET/ {PRINT $1}' "$LOG_FILE" | SORT | UNIQ -C | SORT -NR | HEAD -1: FINDS TOP GET IP.
SIMILAR FOR POST REQUESTS.

10. PATTERNS IN FAILURE REQUESTS:

```
ECHO -E "\N10. PATTERNS IN FAILURE REQUESTS" >> $REPORT_FILE
ECHO "HOURS WITH HIGHEST FAILURE REQUESTS:" >> $REPORT_FILE
AWK '$9 ~ /^[45]/ {PRINT $0}' $LOG_FILE | AWK -F: '{PRINT $2}' | AWK '{PRINT $1}' | SORT | UNIQ -C | SORT -NR | HEAD -3 >>
$REPORT_FILE
```

SAMPLE RUN:

```
HOURS WITH HIGHEST FAILURE REQUESTS:
347 12
328 05
323 11
```

PURPOSE: IDENTIFIES THE MOST ACTIVE IPS FOR GET AND POST REQUESTS SEPARATELY.

COMMANDS:

AWK '/GET/ {PRINT $1}' "$LOG_FILE" | SORT | UNIQ -C | SORT -NR | HEAD -1: FINDS TOP GET IP.

SIMILAR FOR POST REQUESTS.

11. ANALYSIS SUGGESTIONS

```
ECHO -E "\N11. ANALYSIS SUGGESTIONS" >> $REPORT_FILE
ECHO "- INVESTIGATE IPS WITH HIGH FAILURE RATES FOR POTENTIAL MALICIOUS ACTIVITY." >> $REPORT_FILE
ECHO "- SCHEDULE MAINTENANCE DURING LOW-TRAFFIC HOURS (BASED ON HOURLY REQUEST TRENDS)." >> $REPORT_FILE
ECHO "- MONITOR DAYS WITH HIGH FAILURES FOR SERVER OR APPLICATION ISSUES." >> $REPORT_FILE
ECHO "- IMPLEMENT RATE-LIMITING FOR IPS WITH UNUSUALLY HIGH REQUEST COUNTS." >> $REPORT_FILE
```

SAMPLE RUN:

- INVESTIGATE IPS WITH HIGH FAILURE RATES FOR POTENTIAL MALICIOUS ACTIVITY.
- SCHEDULE MAINTENANCE DURING LOW-TRAFFIC HOURS (BASED ON HOURLY REQUEST TRENDS).
- MONITOR DAYS WITH HIGH FAILURES FOR SERVER OR APPLICATION ISSUES.
- IMPLEMENT RATE-LIMITING FOR IPS WITH UNUSUALLY HIGH REQUEST COUNTS.

PURPOSE: OFFERS ACTIONABLE RECOMMENDATIONS BASED ON ANALYSIS.
COMMAND: USES ECHO TO APPEND SUGGESTIONS TO THE REPORT, SUCH AS INVESTIGATING HIGH-FAILURE IPS OR SCHEDULING MAINTENANCE.
CONCLUSION

CONCLUSION

COMPREHENSIVE LOG ANALYSIS:
THE SCRIPT ANALYZES WEB SERVER LOGS TO PROVIDE INSIGHTS INTO TRAFFIC, USER BEHAVIOR, AND POTENTIAL ISSUES.

KEY METRICS:
COUNTS REQUESTS, IDENTIFIES UNIQUE IPS, ANALYZES FAILURES, AND TRACKS PEAK USAGE TIMES.

ACTIONABLE INSIGHTS:
HIGHLIGHTS HIGH-FAILURE IPS AND PEAK HOURS TO GUIDE SECURITY AND PERFORMANCE IMPROVEMENTS.

PROACTIVE RECOMMENDATIONS:
SUGGESTS RATE-LIMITING, MAINTENANCE SCHEDULING, AND MONITORING TO ENHANCE SERVER RELIABILITY.

PROFESSIONAL AND ROBUST:
CORRECTED SYNTAX, IMPROVED READABILITY, AND ADDED INPUT VALIDATION FOR A POLISHED PRESENTATION.