Log Monitoring System – Software Engineering Report

1. Title Page

Project Title: Log Monitoring System – Software Engineering Report
Name: Suyash Pratap Chandel
UID: 23BCS13884

---

2. Introduction

2.1 What is Log Monitoring?

Log monitoring is essential for identifying system issues, tracking errors, and ensuring the stability of software applications. The system automatically scans log files for critical errors and alerts administrators to take necessary actions.

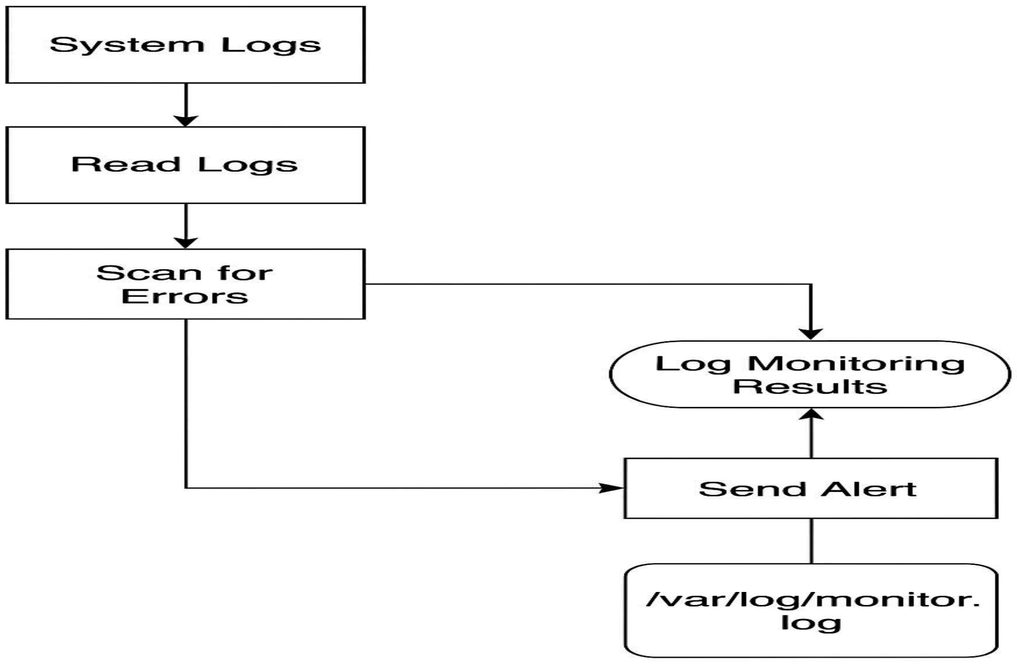
2.2 Project Goals

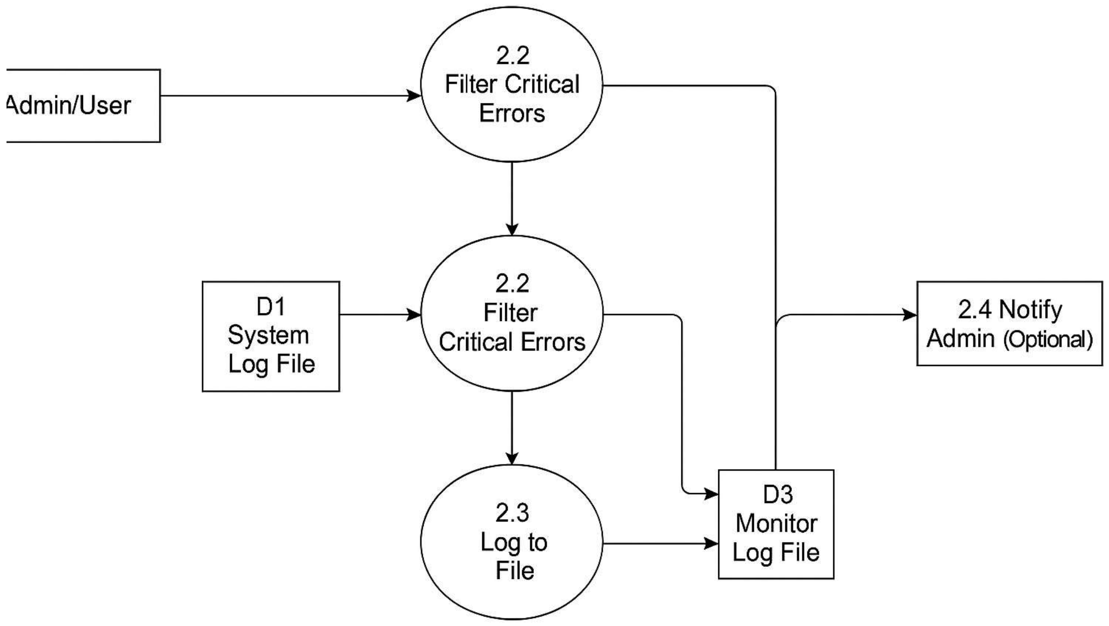
- Automate log monitoring.
- Detect critical errors in system logs.
- Send email alerts upon detection.
- Maintain a monitoring log for record-keeping.

---

3. Software Design

3.1 Data Flow Diagram (DFD) - Level 1 and level 2





Admin/User
2.2 Filter Critical Errors
D1 System Log File
2.2 Filter Critical Errors
2.3 Log to File
D3 Monitor Log File
2.4 Notify Admin (Optional)

3.2 Deployment Architecture

The log monitoring system runs as a shell script on Linux, scheduled via cron jobs. It scans system logs, extracts errors, and sends email alerts.

3.3 Design Principles Used

- **Modularity:** The script uses functions for error scanning and logging.
- **Error Handling:** The script handles errors using trap commands.
- **Automation:** Cron jobs schedule the script execution at fixed intervals.

---

4. Implementation of Shell Script

4.1 Script Logic

1. Read system log files.
2. Scan for critical error messages.
3. Save results to a monitoring log file.
4. Send an email alert if errors are found.

4.2 Code Implementation

```
#!/bin/bash


# Use Windows-style logs and commands safely in Git Bash
```

```
MONITOR_LOG="monitor.log"
TEMP_LOG="temp.log"

echo "Checking system logs..."

# Run wevtutil safely with correct quoting
powershell.exe -Command "wevtutil qe System /c:10 /rd:true /f:text" | grep -i
"error\|failed\|critical" > "$TEMP_LOG"

if [ -s "$TEMP_LOG" ]; then
  echo "$(date) - Critical errors found" >> "$MONITOR_LOG"
  cat "$TEMP_LOG"
else
  echo "$(date) - No critical errors found" >> "$MONITOR_LOG"
fi

rm -f "$TEMP_LOG"
```

---

5. Software Configuration Management (SCM) with Git

5.1 Repository Setup

```
git init
git remote add origin https://github.com/yourusername/log-monitor.git
```

5.2 Branching Strategy

- main: Stable version.
- error-handling: Handles script errors.
- testing: Contains test cases and debugging.

5.3 Git Commit and History

```
git add .
git commit -m "Initial commit - Added log monitor script"
git branch -M main

git push -u origin main

git log --oneline --graph
```

---

6. Testing & Performance Evaluation

6.1 Test Cases

Test Case	Expected Outcome
Run script manually	Detects errors, logs result, and sends alert
No errors in logs	Logs "No critical errors found"
Email alert enabled	Email is sent when an error is detected

6.2 Performance Considerations

- Minimal CPU usage since script runs periodically.
- Can be scaled to monitor multiple log files.

---

7. Risk Analysis & Future Improvements

7.1 Risks

- Email delivery failure.
- Incorrect log file path causing script failure.
- Cron job misconfiguration preventing execution.

7.2 Future Enhancements

- Filtering logs by severity level.
- Integration with cloud-based log monitoring solutions.
- Adding a graphical user interface for real-time monitoring.

---

8. Conclusion

This log monitoring system efficiently detects errors and automates alerts, improving system reliability. Future upgrades will enhance scalability and usability.
