

# Capstone Project Report

---

## 1. Title

Bash Scripting Suite for System Maintenance

## 2. Objective

This project automates Linux system maintenance tasks such as backups, updates, cleanup, and log monitoring using Bash scripting. It is designed to simplify routine administrative work through automation and improve system reliability by scheduling essential maintenance tasks.

## 3. Tools and Technologies

- Programming Language: Bash (v4.0+)
- Platform: Linux (Ubuntu, Debian, Fedora, Arch)
- Tools: tar, apt, dnf, pacman, grep, journalctl, cron, systemd
- Editor: Visual Studio Code
- Version Control: GitHub

## 4. Codes

Below are the implemented Bash scripts used in the project:

```
File Edit Selection View Go Run Terminal Help BashMaintenanceSuite-main

1 #!/usr/bin/env bash
2 # Backup selected sources into timestamped tar.gz archives and prune old ones.
3 set -euo pipefail
4 SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
5 # shellcheck source=utils.sh
6 source "$SCRIPT_DIR/utils.sh"
7 main() {
8     load_env
9     require_root
10    local ts outlog
11    ts="$(timestamp)"
12    outlog="$LOG_DIR/backup_${ts}.log"
13    local sources="${BACKUP_SOURCES:-}"
14    local target="${BACKUP_TARGET:-}"
15    local retention="${RETENTION_DAYS:-7}"
16    if [[ -z "$sources" || -z "$target" ]]; then
17        log ERROR "BACKUP_SOURCES and BACKUP_TARGET must be set in .env"
18        exit 1
19    fi
20    IFS=' ' read -r -a src_arr <<< "$sources"
21    mkdir -p "$target"
22    log INFO "Starting backup -> $target" | tee -a "$outlog"
23    for src in "${src_arr[@]}; do
24        src="$(echo "$src" | xargs)"
25        if [[ -d "$src" || -f "$src" ]]; then
26            base="$(basename "$src")"
27            archive="$target/${base}_${ts}.tar.gz"
28            log INFO "Archiving $src -> $archive" | tee -a "$outlog"
29            tar -czf "$archive" --absolute-names "$src" 2>>"$outlog"
30            log INFO "OK: $archive" | tee -a "$outlog"
31        else
32            log WARN "Skipping missing path: $src" | tee -a "$outlog"
33        fi
34    done
35    # Prune old archives by mtime
36    if [[ "$retention" =~ ^[0-9]+$ ]]; then
37        log INFO "Pruning archives older than $retention days in $target" | tee -a "$outlog"
38        find "$target" -maxdepth 1 -type f -name "*.tar.gz" -mtime +"$retention" -print -delete >>"$outlog" 2>&1 || true
39    fi
40    log INFO "Backup completed." | tee -a "$outlog"
41 }
42 main "$@"
43
```

backup.sh

```
1 #!/usr/bin/env bash
2 # Optional installer: set up cron and systemd timer for log monitoring.
3 set -euo pipefail
4 SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
5 # shellcheck source=utils.sh
6 source "$SCRIPT_DIR/utils.sh"
7 main() {
8     load_env
9     echo "Install options:"
10    echo "1) Install sample crontab (root)"
11    echo "2) Install systemd timer for log monitor (root)"
12    echo "3) Make scripts executable"
13    echo "4) Quit"
14    read -rp "Choose [1-4]: " ans
15    case "$ans" in
16        1)
17            require_root
18            crontab -l 2>/dev/null | cat - "$SCRIPT_DIR/crontab.example" | crontab -
19            log INFO "Cron entries appended."
20            ;;
21        2)
22            require_root
23            svc="bash-maint-log-monitor.service"
24            tim="bash-maint-log-monitor.timer"
25            svc_path="/etc/systemd/system/$svc"
26            tim_path="/etc/systemd/system/$tim"
27            cat > "$svc_path" <<EOF
28            [Unit]
29            Description=Bash Maintenance Suite - Log Monitor
30            [Service]
31            Type=oneshot
32            User=root
33            WorkingDirectory=$SCRIPT_DIR
34            ExecStart=$SCRIPT_DIR/log_monitor.sh
35            Nice=10
36            EOF
37            cat > "$tim_path" <<EOF
38            [Unit]
39            Description=Run Log Monitor every 15 minutes
40            [Timer]
41            OnBootSec=2min
42            OnUnitActiveSec=15min
43            Unit=$svc
44            [Install]
45            WantedBy=timers.target
46            EOF
47            systemctl daemon-reload
48            systemctl enable --now "$tim"
49            log INFO "Systemd timer installed and started."
50            ;;
51        3)
52            chmod +x "$SCRIPT_DIR/*.sh"
53            log INFO "Scripts marked executable."
54            ;;
55        *)
56            echo "Bye."
57            ;;
58    esac
59 }
60 main "$@"
61
```

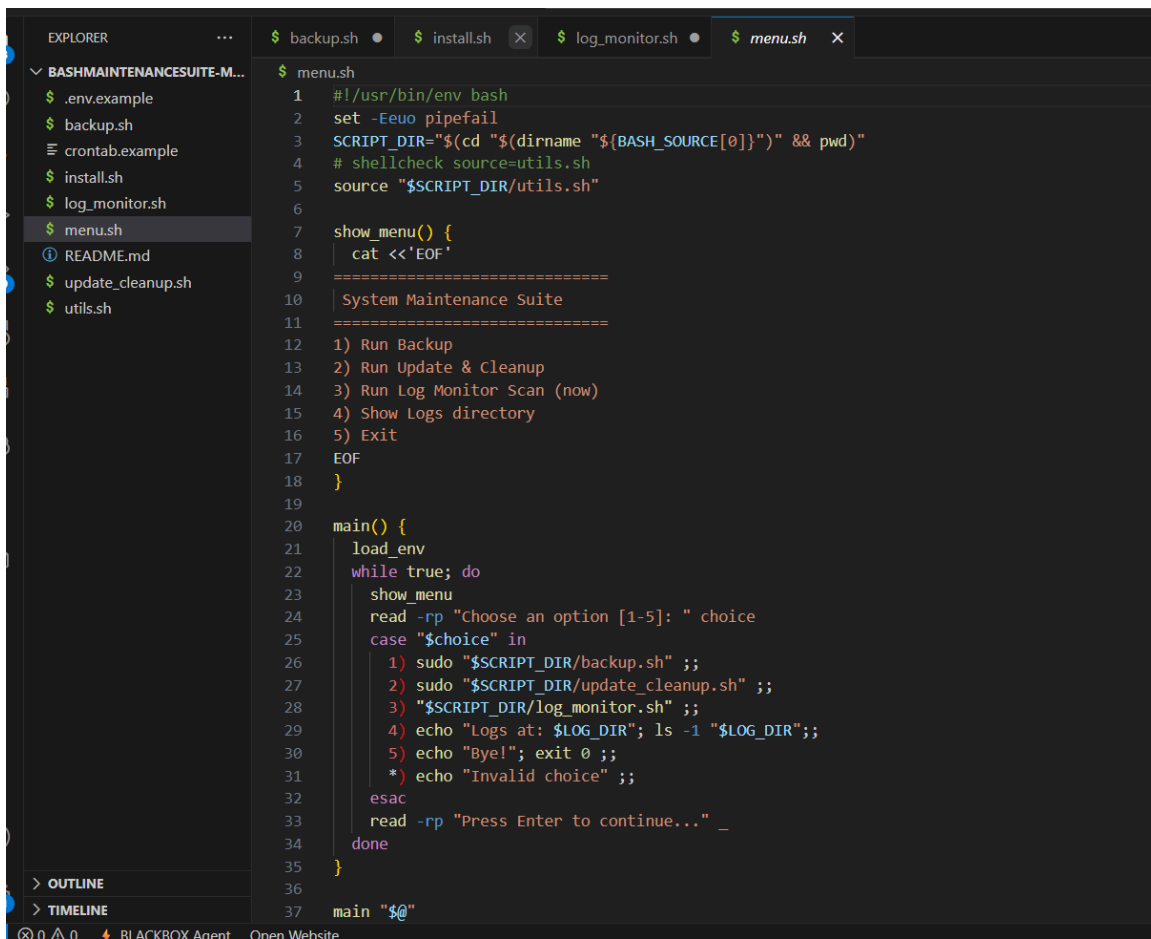
install.sh

```
EXPLORER  ...  $ backup.sh  $ install.sh  $ log_monitor.sh

BASHMAINTENANCESUITE-M...
$ env.example
$ backup.sh
$ cronstab.example
$ install.sh
$ log_monitor.sh
$ menu.sh
① README.md
$ update_cleanup.sh
$ utils.sh

$ log_monitor.sh
1  #!/usr/bin/env bash
2  # Scan logs for keywords within a lookback window and raise alerts. Cron-friendly.
3  set -euo pipefail
4  SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
5  # shellcheck source=utils.sh
6  source "$SCRIPT_DIR/utils.sh"
7  main() {
8      load_env
9      local ts outlog
10     ts="$(timestamp)"
11     outlog="$LOG_DIR/log_monitor_${ts}.log"
12
13     local keywords="${LOG_KEYWORDS:-}"
14     if [[ -z "$keywords" ]]; then
15         log WARN "LOG_KEYWORDS empty; nothing to scan." | tee -a "$outlog"
16         exit 0
17     fi
18     local files="${LOG_FILES:-}"
19     local lookback="${LOG_LOOKBACK_MINUTES:-15}"
20     IFS=', ' read -r -a kw_arr <<< "$keywords"
21     # collect text into buffer
22     tmpfile="$(mktemp)"
23     trap 'rm -f "$tmpfile"' EXIT
24
25     if [[ -n "$files" ]]; then
26         IFS=', ' read -r -a file_arr <<< "$files"
27         for f in "${file_arr[@]}; do
28             fw="$(echo "$f" | xargs)"
29             if [[ -f "$fw" ]]; then
30                 # Tail lines from last X minutes using awk timestamp heuristic (syslog-like: "MMM DD HH:MM:SS")
31                 # Fallback: last 2000 lines
32                 tail -n 2000 "$fw" >> "$tmpfile" 2>/dev/null || true
33             fi
34         done
35     else
36         # Use journalctl for last N minutes
37         if command -v journalctl >/dev/null 2>&1; then
38             journalctl --since "$lookback minutes ago" --no-pager >> "$tmpfile" 2>/dev/null || true
39         elif [[ -f /var/log/syslog ]]; then
40             tail -n 2000 /var/log/syslog >> "$tmpfile" 2>/dev/null || true
41         fi
42     fi
43     local found=0
44     for kw in "${kw_arr[@]}; do
45         kw="$(echo "$kw" | xargs)"
46         if [[ -z "$kw" ]]; then continue; fi
47         if grep -i -E -- "$kw" "$tmpfile" >/dev/null 2>&1; then
48             cnt="$(grep -i -E -- "$kw" "$tmpfile" | wc -l | echo 0)"
49             log WARN "Found '$kw' $cnt in last window." | tee -a "$outlog"
50             send_alert "Log alert: '$kw' detected" "Found $cnt occurrences of '$kw' in the last $lookback minutes."
51             found=$((found+1))
52         fi
53     done
54     if [[ "$found" -eq 1 ]]; then
55         log WARN "Alerts generated." | tee -a "$outlog"
56         exit 2
57     else
58         log INFO "No alerts found." | tee -a "$outlog"
59         exit 0
60     fi
61 }
62 main "$@"
63
```

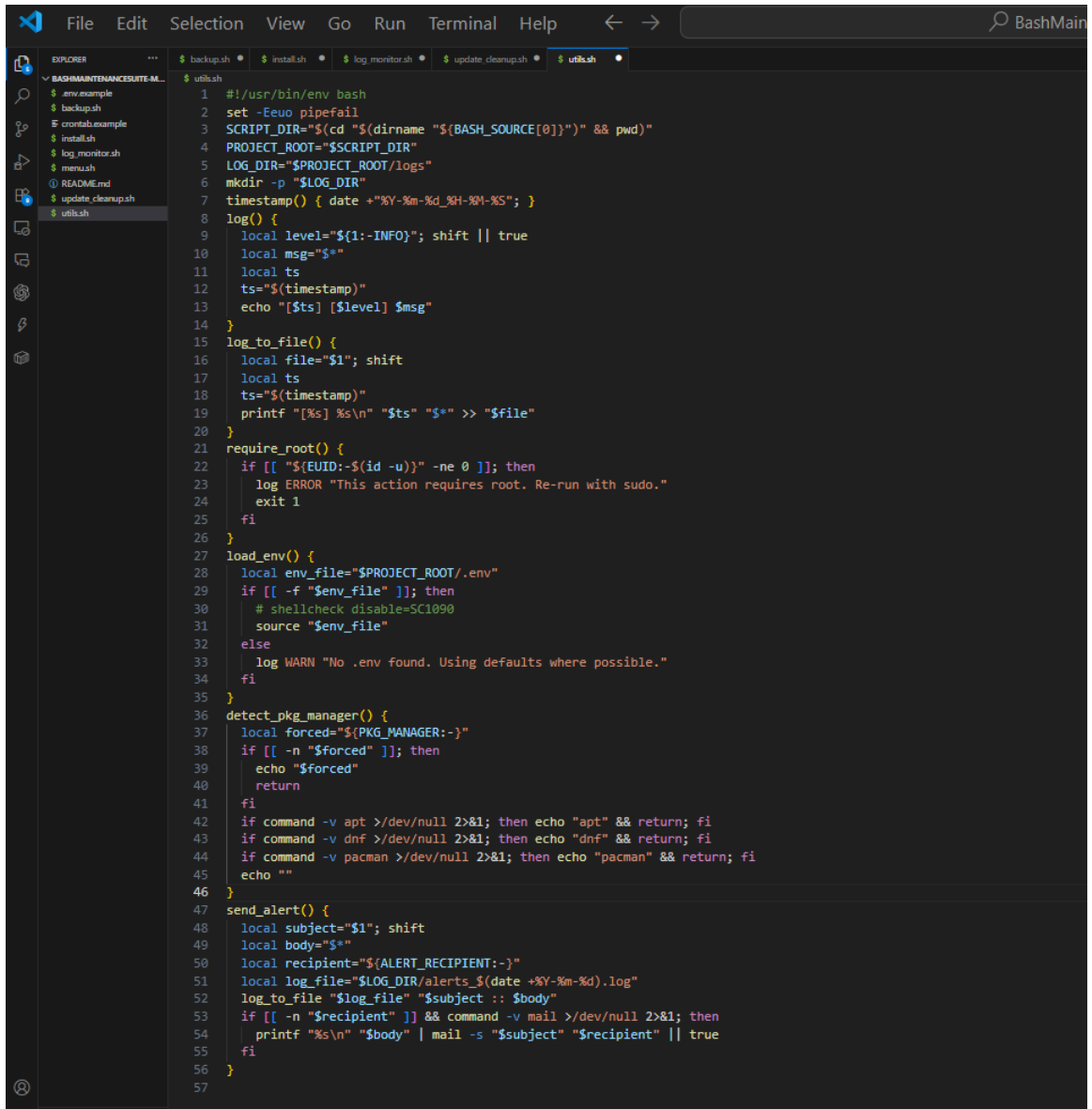
logmonitor.sh



The image shows a terminal window with a file explorer on the left and a code editor on the right. The file explorer shows a directory named 'BASHMAINTENANCESUITE-M...' containing files: .env.example, backup.sh, crontab.example, install.sh, log\_monitor.sh, menu.sh (selected), README.md, update\_cleanup.sh, and utils.sh. The code editor displays the content of menu.sh, which is a shell script for a 'System Maintenance Suite'. The script starts with a shebang, sets error handling, defines SCRIPT\_DIR, and sources utils.sh. It includes a show\_menu() function that prints a menu and a main() function that loops, prompting the user to choose an option from 1 to 5. The options are: 1) Run Backup, 2) Run Update & Cleanup, 3) Run Log Monitor Scan (now), 4) Show Logs directory, and 5) Exit. The script uses sudo to run backup.sh, update\_cleanup.sh, and log\_monitor.sh. It also echoes the log directory and says 'Bye!' before exiting. The script ends with a main "\$@" line.

```
$ menu.sh
1  #!/usr/bin/env bash
2  set -Eeuo pipefail
3  SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
4  # shellcheck source=utils.sh
5  source "$SCRIPT_DIR/utils.sh"
6
7  show_menu() {
8      cat <<'EOF'
9
10     =====
11     System Maintenance Suite
12     =====
13     1) Run Backup
14     2) Run Update & Cleanup
15     3) Run Log Monitor Scan (now)
16     4) Show Logs directory
17     5) Exit
18     EOF
19 }
20
21 main() {
22     load_env
23     while true; do
24         show_menu
25         read -rp "Choose an option [1-5]: " choice
26         case "$choice" in
27             1) sudo "$SCRIPT_DIR/backup.sh" ;;
28             2) sudo "$SCRIPT_DIR/update_cleanup.sh" ;;
29             3) "$SCRIPT_DIR/log_monitor.sh" ;;
30             4) echo "Logs at: $LOG_DIR"; ls -l "$LOG_DIR";;
31             5) echo "Bye!"; exit 0 ;;
32             *) echo "Invalid choice" ;;
33         esac
34         read -rp "Press Enter to continue..." _
35     done
36 }
37
38 main "$@"
```

menu.sh



The image shows a code editor window with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The Explorer sidebar on the left shows a project structure with files like env.example, backup.sh, cronjob.example, install.sh, log\_monitor.sh, menu.sh, README.md, update\_cleanup.sh, and ultra.sh. The main editor area displays the content of ultra.sh, which is a Bash script. The script includes functions for logging, requiring root, loading environment files, detecting package managers, and sending alerts. The script is numbered from 1 to 57.

```
1 #!/usr/bin/env bash
2 set -euo pipefail
3 SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
4 PROJECT_ROOT="$SCRIPT_DIR"
5 LOG_DIR="$PROJECT_ROOT/logs"
6 mkdir -p "$LOG_DIR"
7 timestamp() { date +"%Y-%m-%d_%H-%M-%S"; }
8 log() {
9     local level="${1:-INFO}"; shift || true
10    local msg="$*"
11    local ts
12    ts="$(timestamp)"
13    echo "[$ts] [$level] $msg"
14 }
15 log_to_file() {
16     local file="$1"; shift
17     local ts
18     ts="$(timestamp)"
19     printf "[%s] %s\n" "$ts" "$*" >> "$file"
20 }
21 require_root() {
22     if [[ "${EUID:-$(id -u)}" -ne 0 ]]; then
23         log ERROR "This action requires root. Re-run with sudo."
24         exit 1
25     fi
26 }
27 load_env() {
28     local env_file="$PROJECT_ROOT/.env"
29     if [[ -f "$env_file" ]]; then
30         # shellcheck disable=SC1090
31         source "$env_file"
32     else
33         log WARN "No .env found. Using defaults where possible."
34     fi
35 }
36 detect_pkg_manager() {
37     local forced="${PKG_MANAGER:-}"
38     if [[ -n "$forced" ]]; then
39         echo "$forced"
40         return
41     fi
42     if command -v apt >/dev/null 2>&1; then echo "apt" && return; fi
43     if command -v dnf >/dev/null 2>&1; then echo "dnf" && return; fi
44     if command -v pacman >/dev/null 2>&1; then echo "pacman" && return; fi
45     echo ""
46 }
47 send_alert() {
48     local subject="$1"; shift
49     local body="$*"
50     local recipient="${ALERT_RECIPIENT:-}"
51     local log_file="$LOG_DIR/alerts_$(date +"%Y-%m-%d").log"
52     log_to_file "$log_file" "$subject :: $body"
53     if [[ -n "$recipient" ]] && command -v mail >/dev/null 2>&1; then
54         printf "%s\n" "$body" | mail -s "$subject" "$recipient" || true
55     fi
56 }
57
```

ultra.sh

```

BASHMAINTENANCESUITE-M... $ update_cleanup.sh
1  #!/usr/bin/env bash
2  # Update packages and clean caches. Supports apt, dnf, pacman.
3  set -Eeuo pipefail
4  SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
5  # shellcheck source=utils.sh
6  source "$SCRIPT_DIR/utils.sh"
7  main() {
8      load_env
9      require_root
10     local pm outlog ts
11     ts="$(timestamp)"
12     outlog="$LOG_DIR/update_cleanup_${ts}.log"
13     pm="$(detect_pkg_manager)"
14     log INFO "Detected package manager: ${pm:-unknown}" | tee -a "$outlog"
15     case "$pm" in
16         apt)
17             export DEBIAN_FRONTEND=noninteractive
18             apt update | tee -a "$outlog"
19             apt -y upgrade | tee -a "$outlog"
20             apt -y autoremove | tee -a "$outlog"
21             apt -y autoclean | tee -a "$outlog"
22             ;;
23         dnf)
24             dnf -y upgrade | tee -a "$outlog"
25             dnf -y autoremove | tee -a "$outlog" || true
26             dnf -y clean all | tee -a "$outlog"
27             ;;
28         pacman)
29             pacman -Syu --noconfirm | tee -a "$outlog"
30             paccache -r -k 3 2>>"$outlog" || true # keep last 3 versions
31             ;;
32         *)
33             log ERROR "Unsupported or undetected package manager." | tee -a "$outlog"
34             ;;
35     esac
36     # General cleanup: journald & tmp
37     if command -v journalctl >/dev/null 2>&1; then
38         journalctl --vacuum-time=14d >>"$outlog" 2>&1 || true
39     fi
40     find /tmp -type f -mtime +7 -delete 2>>"$outlog" || true
41     log INFO "Update & cleanup completed." | tee -a "$outlog"
42 }
43 main "$@"
44

```

updatecleanup.sh

## 5. Full Execution Screenshots

The following screenshots demonstrate the successful execution of the scripts and their outputs:

## 2 Backup File Output

```
ALI File Actions Edit View Help

(akshya@kali)~[/Documents/Wipro Project/BashMaintenanceSuite-main]
$ cd BashMaintenanceSuite
$ cp .env.example .env
# Edit .env to match your system
chmod +x *.sh
./menu.sh
cd: no such file or directory: BashMaintenanceSuite

System Maintenance Suite
=====
1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
Choose an option [1-5]: 1
[sudo] password for akshya:
[2025-11-08_15-48-31] [INFO] Starting backup → /var/backups
[2025-11-08_15-48-32] [INFO] Archiving /etc → /var/backups/etc_2025-11-08_15-48-31.tar.gz
[2025-11-08_15-48-38] [INFO] OK: /var/backups/etc_2025-11-08_15-48-31.tar.gz
[2025-11-08_15-48-38] [INFO] Archiving /home → /var/backups/home_2025-11-08_15-48-31.tar.gz
[2025-11-08_15-48-46] [INFO] OK: /var/backups/home_2025-11-08_15-48-31.tar.gz
[2025-11-08_15-48-46] [INFO] Pruning archives older than 7 days in /var/backups
ps
[2025-11-08_15-48-46] [INFO] Backup completed.
Press Enter to continue ...
```

## 2 System Update and Cleanup Output

```
Press Enter to continue ...

System Maintenance Suite
=====
1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
Choose an option [1-5]: 2
[2025-11-08_15-48-58] [INFO] Detected package manager: apt

WARNING: apt does not have a stable CLI interface. Use with caution in script
s.

Ign:1 http://http.kali.org/kali kali-rolling InRelease
Ign:1 http://http.kali.org/kali kali-rolling InRelease
Ign:1 http://http.kali.org/kali kali-rolling InRelease
Err:1 http://http.kali.org/kali kali-rolling InRelease
Temporary failure resolving 'http.kali.org'
Reading package lists...
Building dependency tree...
Reading state information...
All packages are up to date.
Warning: Failed to fetch http://http.kali.org/kali/dists/kali-rolling/InRelease Temporary failure resolving 'http.kali.org'
Warning: Some index files failed to download. They have been ignored, or old ones used instead.

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
Calculating upgrade...
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
Summary:
```

```
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
[2025-11-08_15-50-20] [INFO] Update & cleanup completed.
Press Enter to continue ...

System Maintenance Suite
=====
1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
Choose an option [1-5]: 3
[2025-11-08_15-50-27] [INFO] No alerts found.
Press Enter to continue ...
```



## ❏ Log Monitoring Output

```
File Actions Edit View Help
Press Enter to continue...

System Maintenance Suite

1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
Choose an option [1-5]: 3
[2025-11-08_15-50-27] [INFO] No alerts found.
Press Enter to continue...
```

## ❏ Show log directory

```
System Maintenance Suite

1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
Choose an option [1-5]: 4
Logs at: /home/akshya/Documents/Wipro Project/BashMaintenanceSuite-main/logs
backup_2025-11-08_15-48-31.log
log_monitor_2025-11-08_15-50-27.log
update_cleanup_2025-11-08_15-48-58.log
Press Enter to continue...
```

## ❏ Exit

```
System Maintenance Suite

1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
Choose an option [1-5]: 5
Bye!
```

## 6. GitHub Repository

Link- “ <https://github.com/akshya44/Linux-System-Maintenance-Suite.git>”

## 7. Conclusion

The Bash Scripting Suite for System Maintenance provides a practical and efficient automation solution for routine Linux system tasks. It improves system uptime, reduces manual effort, and demonstrates real-world DevOps and shell scripting skills.