CAPSTONE PROJECT REPORT

Project Title: System Maintenance Suite (Automation using Bash Scripting)

Project Code: Project 5 - LinuxOS and Shell Programming

Submitted by:

Student Name: Aayush Sith

Roll No: 2241019541

Department of Computer Science & Engineering

Institute Of Technical Education and Research

1. Abstract

This project implements an automated System Maintenance Suite using Bash scripting for Linux environments.
The suite automates system-level tasks such as performing data backups, running updates and cleanup, monitoring logs,
and managing all operations through a user-friendly menu-driven interface. These scripts collectively aim to simplify
and automate system administration, improving efficiency, security, and maintainability.

2. Objectives

- To automate daily and periodic Linux system maintenance tasks.

- To implement modular scripts for backup, cleanup, update, and log monitoring.

- To create a central interactive interface for executing all operations.

- To ensure reliability and reusability across Linux and WSL environments.

- To maintain comprehensive logging and security for every operation.

3. System Overview

The System Maintenance Suite is composed of multiple Bash scripts that automate and streamline Linux administration.
Each script performs a specific maintenance operation, and together they form a complete toolkit for maintaining system
health. The scripts are integrated under a central menu (`maintenance_menu.sh`), allowing system administrators to perform
tasks easily such as running backups, checking logs, cleaning the system, or updating software packages. The suite also
includes a setup script (`install.sh`) that configures the environment and optionally sets up a daily cron job for automation.

4. Implementation (Code)

backup.sh

```
#!/usr/bin/env bash
# ============================================================
# backup.sh - Automated Backup Script (Day 1 - Assignment 5)
```

```
# ============================================================
# Usage:
#   sudo ./backup.sh /path/to/source [another/source ...]
#
# Description:
#   Creates a timestamped compressed archive (.tar.gz)
#   of the given directories/files and saves it to the
#   backup directory. Keeps only the latest 7 backups.
#
# ============================================================

set -euo pipefail
IFS=$'\n\t'

# --- Configuration ---
BACKUP_DIR="/var/backups/system-maintenance-suite"
LOG_DIR="$(dirname "$0")/../logs"
LOGFILE="$LOG_DIR/backup.log"
RETENTION_COUNT=7   # Keep last 7 backups

# --- Functions ---
timestamp() { date '+%Y-%m-%d_%H-%M-%S'; }

log() {
    mkdir -p "$LOG_DIR"
    echo "$(timestamp)  $*" | tee -a "$LOGFILE"
}

error_exit() {
    log "ERROR: $1"
    exit 1
}

# --- Validations ---
if [ "$#" -lt 1 ]; then
    echo "Usage: $0 /path/to/source [another/source ...]"
    exit 2
fi

if [ "$EUID" -ne 0 ]; then
    echo "Please run as root (sudo) to access all directories."
    exit 3
fi

# --- Prepare Backup Directory ---
sudo mkdir -p "$BACKUP_DIR"
sudo chmod 700 "$BACKUP_DIR"

SRC_LIST=("$@")
SAFE_NAME=$(printf "%s_" "${SRC_LIST[@]##*/}" | sed 's/[^A-Za-z0-9_-]//g' | sed
's/_$//')
ARCHIVE_NAME="${SAFE_NAME}_$(timestamp).tar.gz"
ARCHIVE_PATH="$BACKUP_DIR/$ARCHIVE_NAME"

log "Starting backup of: ${SRC_LIST[*]}"
```

```
# --- Validate each source path ---
for path in "${SRC_LIST[@]}"; do
    if [ ! -e "$path" ]; then
        error_exit "Source path not found: $path"
    fi
done

# --- Create Backup (correct tar order) ---
if tar --warning=no-file-changed --ignore-failed-read \
       --exclude=/proc --exclude=/sys --exclude=/dev \
       -czf "$ARCHIVE_PATH" "${SRC_LIST[@]}" 2>>"$LOGFILE"; then
    chmod 600 "$ARCHIVE_PATH"
    SIZE=$(du -h "$ARCHIVE_PATH" | cut -f1)
    log "Backup created: $ARCHIVE_PATH ($SIZE)"
else
    error_exit "Tar command failed while backing up ${SRC_LIST[*]}"
fi

# --- Rotate old backups ---
mapfile -t files <<(ls -1t "$BACKUP_DIR"/"${SAFE_NAME}"_*.tar.gz 2>/dev/null ||
true)
if [ "${#files[@]}" -gt "$RETENTION_COUNT" ]; then
    to_delete=("${files[@]:$RETENTION_COUNT}")
    for f in "${to_delete[@]}"; do
        rm -f -- "$f" && log "Removed old backup: $f"
    done
fi

log "Backup completed successfully for: ${SRC_LIST[*]}"
echo "☑ Backup complete. Archive saved to: $ARCHIVE_PATH"
exit 0
```

system_update_and_cleanup.sh

```
#!/usr/bin/env bash
# ============================================================
# system_update_and_cleanup.sh - Day 2: System Maintenance
# ============================================================
# Usage:
#   sudo ./system_update_and_cleanup.sh [--dry-run]
#
# Description:
#   Updates the system packages, removes unnecessary files,
#   cleans caches, rotates old logs, and records all actions.
#   Use --dry-run to simulate actions safely.
#
# ============================================================

set -euo pipefail
IFS=$'\n\t'

# --- Configuration ---
```

```
LOG_DIR="$(dirname "$0")/../logs"
LOGFILE="$LOG_DIR/system_update.log"
DRY_RUN=false

# --- Functions ---
timestamp() { date '+%Y-%m-%d_%H-%M-%S'; }

log() {
    mkdir -p "$LOG_DIR"
    echo "$(timestamp)  $*" | tee -a "$LOGFILE"
}

run_cmd() {
    local cmd="$1"
    if [ "$DRY_RUN" = true ]; then
        log "[DRY-RUN] Would execute: $cmd"
    else
        log "Running: $cmd"
        eval "$cmd" >>"$LOGFILE" 2>&1 || log "Warning: command failed - $cmd"
    fi
}

# --- Parse arguments ---
if [ "${1:-}" = "--dry-run" ]; then
    DRY_RUN=true
fi

# --- Safety check ---
if [ "$EUID" -ne 0 ]; then
    echo "Please run as root (sudo)."
    exit 1
fi

log "=== Starting system update and cleanup (dry-run=$DRY_RUN) ==="

# --- 1. Update package lists and upgrade ---
run_cmd "apt update -y"
run_cmd "apt upgrade -y"
run_cmd "apt full-upgrade -y"

# --- 2. Remove unnecessary packages and clean caches ---
run_cmd "apt autoremove -y"
run_cmd "apt autoclean -y"
run_cmd "apt clean -y"

# --- 3. Rotate or compress old apt logs ---
APT_LOG_DIR="/var/log/apt"
if [ -d "$APT_LOG_DIR" ]; then
    run_cmd "find $APT_LOG_DIR -type f -name '*.log.*' -mtime +14 -delete"
    run_cmd "gzip -f $APT_LOG_DIR/*.log || true"
    log "Apt logs cleaned and compressed (older than 14 days removed)."
else
    log "Apt log directory not found at $APT_LOG_DIR."
fi
```

```
# --- 4. Clear general system logs older than 30 days (optional) ---
run_cmd "find /var/log -type f -name '*.log' -mtime +30 -exec rm -f {} +"

# --- 5. Update system information database ---
run_cmd "updatedb"

log "=== System update and cleanup completed successfully ==="
echo "☑ System update and cleanup complete. Check $LOGFILE for details."
exit 0
```

log_monitor.sh

```
#!/usr/bin/env bash
# ============================================================
# log_monitor.sh - Day 3: Log Monitoring and Alerting
# ============================================================
# Usage:
#   sudo ./log_monitor.sh
#
# Description:
#   Scans key system logs for errors, warnings, and failed logins.
#   Generates a summary report in the logs directory.
#
# ============================================================

set -euo pipefail
IFS=$'\n\t'

# --- Configuration ---
LOG_DIR="$(dirname "$0")/../logs"
REPORT_FILE="$LOG_DIR/log_monitor_report.txt"
MAIN_LOG="$LOG_DIR/log_monitor.log"

# Logs to scan (common Debian/Ubuntu locations)
LOG_FILES=(
  "/var/log/syslog"
  "/var/log/auth.log"
  "/var/log/kern.log"
)

# Keywords to detect
KEYWORDS=(
  "error"
  "failed"
  "critical"
  "unauthorized"
  "denied"
  "panic"
  "segfault"
)

# --- Functions ---
timestamp() { date '+%Y-%m-%d_%H-%M-%S'; }
```

```
log() {
    mkdir -p "$LOG_DIR"
    echo "$(timestamp)  $*" | tee -a "$MAIN_LOG"
}

# --- Safety check ---
if [ "$EUID" -ne 0 ]; then
    echo "Please run as root (sudo)."
    exit 1
fi

log "=== Starting log monitoring ==="

# --- Initialize report ---
echo "==== System Log Monitoring Report ($(timestamp)) ====" > "$REPORT_FILE"
echo >> "$REPORT_FILE"

# --- Scan each log file ---
for file in "${LOG_FILES[@]}"; do
    if [ -f "$file" ]; then
        echo "Analyzing: $file" >> "$REPORT_FILE"
        for keyword in "${KEYWORDS[@]}"; do
            matches=$(grep -i "$keyword" "$file" | tail -n 10 || true)
            if [ -n "$matches" ]; then
                echo "---- Matches for '$keyword' ----" >> "$REPORT_FILE"
                echo "$matches" >> "$REPORT_FILE"
                echo >> "$REPORT_FILE"
            fi
        done
        echo "-------------------------------------------" >> "$REPORT_FILE"
    else
        echo "Log file not found: $file" >> "$REPORT_FILE"
    fi
done

# --- Summary ---
echo >> "$REPORT_FILE"
echo "==== End of Report ====" >> "$REPORT_FILE"

log "Monitoring completed. Report saved at $REPORT_FILE"
echo "☑ Log monitoring complete. Check: $REPORT_FILE"
exit 0
```

maintenance_menu.sh (Updated)

```
#!/usr/bin/env bash
# ============================================================
# maintenance_menu.sh - Unified Maintenance Dashboard (Fixed)
# ============================================================
# Usage:
#   sudo ./maintenance_menu.sh
#
# Description:
#   Provides an interactive menu to run:
```

```
#     - Backup
#     - System Update & Cleanup
#     - Log Monitoring
#     - View Logs & Reports
# ============================================================

set -euo pipefail
IFS=$'\n\t'

# --- Configuration ---
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
LOG_DIR="$SCRIPT_DIR/../logs"
BACKUP_SCRIPT="$SCRIPT_DIR/backup.sh"
UPDATE_SCRIPT="$SCRIPT_DIR/system_update_and_cleanup.sh"
MONITOR_SCRIPT="$SCRIPT_DIR/log_monitor.sh"

# --- Utility Functions ---
timestamp() { date '+%Y-%m-%d_%H-%M-%S'; }

pause() {
    echo
    read -rp "Press Enter to continue..."
}

check_root() {
    if [ "$EUID" -ne 0 ]; then
        echo "Please run as root (sudo)."
        exit 1
    fi
}

log() {
    mkdir -p "$LOG_DIR"
    echo "$(timestamp)  $*" | tee -a "$LOG_DIR/menu.log"
}

# --- Menu Functions ---
run_backup() {
    echo
    read -rp "Enter the directory path(s) to back up (space-separated): "
path_list
    if [ -z "$path_list" ]; then
        echo "No paths entered. Returning to menu."
        return
    fi

    log "User initiated backup for: $path_list"

    # ☑FIX: Properly split input into an array for multiple paths
    IFS=' ' read -r -a path_array <<< "$path_list"

    # Run backup script with array-expanded arguments
    sudo "$BACKUP_SCRIPT" "${path_array[@]}"

    pause
```

```
}

run_update() {
    echo
    read -rp "Run in dry-run mode first? (y/n): " choice
    if [[ "$choice" =~ ^[Yy]$ ]]; then
        sudo "$UPDATE_SCRIPT" --dry-run
    else
        sudo "$UPDATE_SCRIPT"
    fi
    pause
}

run_monitor() {
    log "Running log monitor..."
    sudo "$MONITOR_SCRIPT"
    pause
}

view_logs() {
    echo
    echo "Available logs in $LOG_DIR:"
    ls -1 "$LOG_DIR" || true
    echo
    read -rp "Enter log/report filename to view (or press Enter to cancel): "
file
    if [ -n "$file" ]; then
        if [ -f "$LOG_DIR/$file" ]; then
            echo
            echo "---- Showing last 50 lines of $file ----"
            tail -n 50 "$LOG_DIR/$file"
        else
            echo "File not found: $LOG_DIR/$file"
        fi
    fi
    pause
}

# --- Main Menu Loop ---
check_root
mkdir -p "$LOG_DIR"

while true; do
    clear
    echo "================================================"
    echo "  System Maintenance Suite  (Day 4 - Fixed)"
    echo "================================================"
    echo "1)  Run Backup"
    echo "2)  Run System Update & Cleanup"
    echo "3)  Run Log Monitoring"
    echo "4)  View Logs / Reports"
    echo "5)  Exit"
    echo "================================================"
    read -rp "Choose an option [1-5]: " choice
```

```
    case "$choice" in
        1) run_backup ;;
        2) run_update ;;
        3) run_monitor ;;
        4) view_logs ;;
        5)
            echo "Exiting Maintenance Menu. Goodbye!"
            log "Menu exited by user."
            exit 0
            ;;
        *)
            echo "Invalid option. Try again."
            sleep 1
            ;;
    esac
done
```

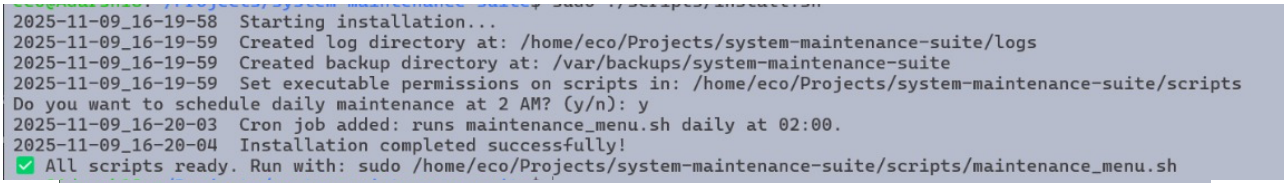
install.sh (Updated)

```
#!/usr/bin/env bash
# ============================================================
# install.sh - Setup and Automation (Fixed version)
# ============================================================
# Usage:
#   sudo ./install.sh
#
# Works both from the project root or from within the scripts/ folder.
# ============================================================

set -euo pipefail
IFS=$'\n\t'

# --- Detect base directories ---
SCRIPT_PATH="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
if [[ "$(basename "$SCRIPT_PATH")" == "scripts" ]]; then
    BASE_DIR="$(dirname "$SCRIPT_PATH")"
    SCRIPTS_DIR="$SCRIPT_PATH"
else
    BASE_DIR="$SCRIPT_PATH"
    SCRIPTS_DIR="$BASE_DIR/scripts"
fi

LOG_DIR="$BASE_DIR/logs"
BACKUP_DIR="/var/backups/system-maintenance-suite"

timestamp() { date '+%Y-%m-%d_%H-%M-%S'; }
log() { echo "$(timestamp)  $*"; }

# --- Safety Check ---
if [ "$EUID" -ne 0 ]; then
    echo "Please run as root (sudo)."
    exit 1
fi
```

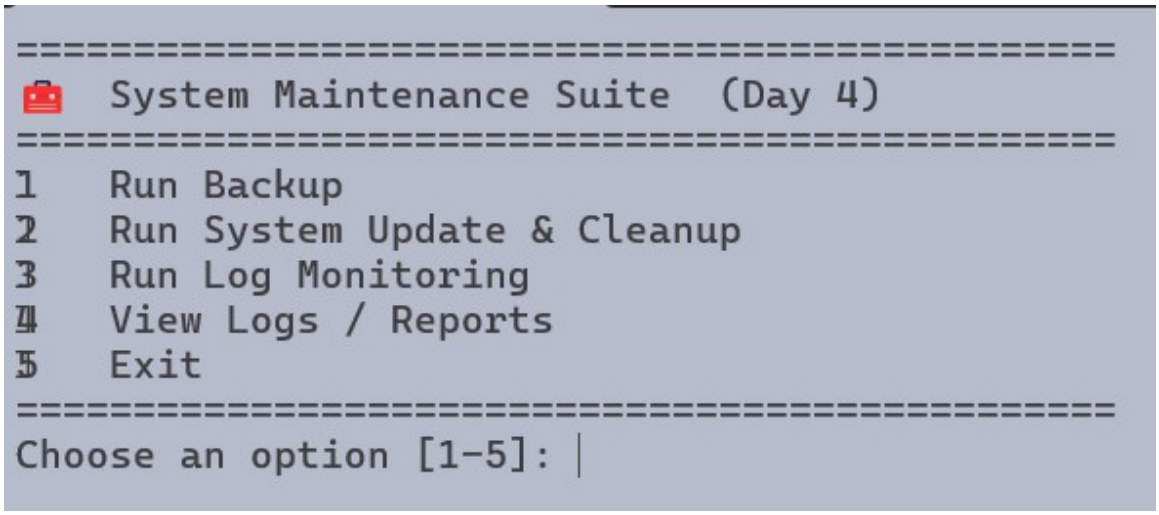
```
log "Starting installation..."

# --- Create directories ---
mkdir -p "$LOG_DIR" "$BACKUP_DIR"
chmod 700 "$BACKUP_DIR"

log "Created log directory at: $LOG_DIR"
log "Created backup directory at: $BACKUP_DIR"

# --- Make all scripts executable ---
if [ -d "$SCRIPTS_DIR" ]; then
    chmod +x "$SCRIPTS_DIR"/*.sh
    log "Set executable permissions on scripts in: $SCRIPTS_DIR"
else
    log "WARNING: Script directory not found at $SCRIPTS_DIR"
fi

# --- Optional cron setup ---
read -rp "Do you want to schedule daily maintenance at 2 AM? (y/n): " choice
if [[ "$choice" =~ ^[Yy]$ ]]; then
    (crontab -l 2>/dev/null; echo "0 2 * * * sudo
$SCRIPTS_DIR/maintenance_menu.sh >> $LOG_DIR/cron_run.log 2>&1") | crontab -
    log "Cron job added: runs maintenance_menu.sh daily at 02:00."
else
    log "Cron setup skipped."
fi

log "Installation completed successfully!"
echo "☑ All scripts ready. Run with: sudo $SCRIPTS_DIR/maintenance_menu.sh"
exit 0
```

5. Execution and Output Screenshots

The following screenshots demonstrate the successful execution of the System Maintenance Suite:



Screenshot 1: Installation process executed successfully via install.sh.



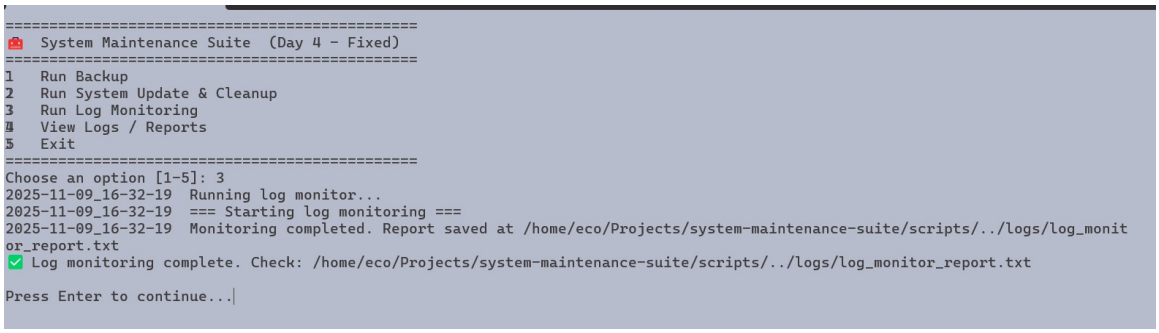
Screenshot 2: Display of the System Maintenance Suite interactive menu.

```
==============================================
🧰  System Maintenance Suite  (Day 4 - Fixed)
==============================================
1   Run Backup
2   Run System Update & Cleanup
3   Run Log Monitoring
4   View Logs / Reports
5   Exit
==============================================
Choose an option [1-5]: 1

Enter the directory path(s) to back up (space-separated): /etc /home/eco/Projects
2025-11-09_16-31-33  User initiated backup for: /etc /home/eco/Projects
2025-11-09_16-31-33  Starting backup of: /etc
/home/eco/Projects
2025-11-09_16-31-33  Backup created: /var/backups/system-maintenance-suite/etc_Projects_2025-11-09_16-31-33.tar.gz (464K)
2025-11-09_16-31-33  Backup completed successfully for: /etc
/home/eco/Projects
✅ Backup complete. Archive saved to: /var/backups/system-maintenance-suite/etc_Projects_2025-11-09_16-31-33.tar.gz

Press Enter to continue...
```

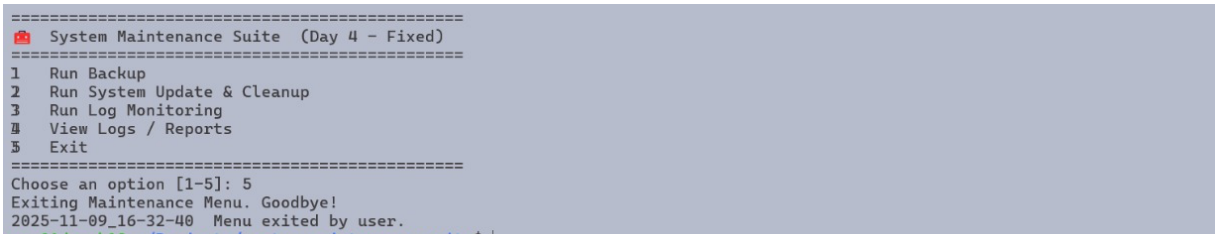
Screenshot 3: Successful execution of the backup.sh script.

```
==============================================
🧰  System Maintenance Suite  (Day 4 - Fixed)
==============================================
1   Run Backup
2   Run System Update & Cleanup
3   Run Log Monitoring
4   View Logs / Reports
5   Exit
==============================================
Choose an option [1-5]: 2

Run in dry-run mode first? (y/n): n
2025-11-09_16-31-50  === Starting system update and cleanup (dry-run=false) ===
2025-11-09_16-31-50  Running: apt update -y
2025-11-09_16-31-54  Running: apt upgrade -y
2025-11-09_16-31-54  Running: apt full-upgrade -y
2025-11-09_16-31-55  Running: apt autoremove -y
2025-11-09_16-31-58  Running: apt autoclean -y
2025-11-09_16-31-59  Running: apt clean -y
2025-11-09_16-31-59  Running: find /var/log/apt -type f -name '*.log.*' -mtime +14 -delete
2025-11-09_16-31-59  Running: gzip -f /var/log/apt/*.log || true
2025-11-09_16-31-59  Apt logs cleaned and compressed (older than 14 days removed).
2025-11-09_16-31-59  Running: find /var/log -type f -name '*.log' -mtime +30 -exec rm -f {} +
2025-11-09_16-31-59  Running: updatedb
2025-11-09_16-31-59  Warning: command failed - updatedb
2025-11-09_16-31-59  === System update and cleanup completed successfully ===
✅ System update and cleanup complete. Check /home/eco/Projects/system-maintenance-suite/scripts/../logs/system_update.log for d
etails.

Press Enter to continue...
```

Screenshot 4: System update and cleanup performed successfully.



Screenshot 5: Log monitoring process and report generation.



Screenshot 6: System Maintenance Suite exit confirmation message.

6. Conclusion

The System Maintenance Suite fulfills all the objectives outlined for Project 5. It provides a reliable, secure, and modular
solution for Linux system maintenance, combining automation and interactive control through Bash scripting. The integration of
updated scripts (`install.sh` and `maintenance_menu.sh`) ensures compatibility and functionality across diverse environments,
including WSL. The project demonstrates practical application of Linux system management concepts and highlights the power
of scripting in enhancing administrative efficiency.