PROJECT REPORT

Topic: Bash Scripting Suite for System Maintenance

Submitted By:

Name: NIHAR KUMAR DASH

Reg. No.: 2241004167

Batch – 11

Institute of Technical Education and Research

SIKSHA 'O' ANUSANDHAN (DEEMED TO BE UNIVERSITY)

Bhubaneswar, Odisha, 751030

INTRODUCTION:

Objective: Write a suite of Bash scripts to automate system maintenance tasks such as backup, system updates, and log monitoring

This project, the "Bash Scripting Suite for System Maintenance," is designed to simplify and automate common administrative tasks on a Linux system. It addresses the repetitive nature of system upkeep by providing a set of modular Bash scripts combined into one central, menu-driven application. The primary goal is to improve efficiency by allowing a user to perform system backups, run software updates, and monitor critical logs by selecting simple menu options. The suite also ensures accountability and aids troubleshooting by logging all actions and their outputs to a central log file. By leveraging powerful, built-t-in Linux commands like tar for backups, apt for package management, and grep for log analysis, the suite provides a robust solution without external dependencies. This approach not only saves significant time but also reduces the potential for human error in performing routine maintenance.

IMPLEMENTATION PLAN:

This project was implemented in five stages. The core functions were first built as modular scripts (Days 1-3), then integrated into a user-friendly suite (Day 4), and finally, hardened with error handling and logging (Day 5).

---

1. Day 1: Automated System Backup Script

- Developed backup.sh to automate the backup of a specified source directory.
- Used the tar utility to create compressed .tar.gz archives.
- Integrated the date command to generate unique, timestamped filenames, preventing overwrites.
- Defined SOURCE_DIR and BACKUP_DIR as variables for easy configuration.

---

2. Day 2: System Update and Cleanup Script

- Created maintenance.sh to automate routine system maintenance using the APT package manager.
- The script executes a sequence of commands: apt update, apt upgrade -y (installs updates), apt auto remove -y (removes old dependencies), and apt clean (clears the package cache).
- Added set -e to ensure the script stops immediately if any command fails.

---

3. Day 3: Log Monitoring Script

- Developed log_monitor.sh to scan system logs for potential issues.
- Used the grep command as the core tool to search a target log file (e.g., /var/log/auth.log) for a specific KEYWORD (e.g., "Failed").

- The script reports all matching lines to provide context for alerts or confirms a clean scan.

---

4. Day 4: Maintenance Suite Menu

- Built a master script, main.sh, to act as a central user interface.
- A while true loop was used to continuously display the menu options.
- A case statement was implemented to read the user's input and execute the corresponding script (Backup, Maintenance, or Log Monitor).

---

5. Day 5: Error Handling and Logging

- This final stage involved updating all scripts to make the suite robust and auditable.
- Error Handling: Added validation checks (e.g., if [! -d "$SOURCE_DIR"]) to ensure files and directories exist before use.
- Logging: Modified main.sh to use the tee -a command. This pipes all script output to both the user's screen and a central log file (maintenance_suite.log) for future review.

Bash Script:

backup.sh:

```
#!/bin/bash
set -e # Exit immediately if a command fails

# --- Configuration ---
SOURCE_DIR="/home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting"
BACKUP_DIR="/home/kali/Documents/Backup/my project"
FILENAME="backup_$(date   +'%Y-%m-%d_%H-%M-%S').tar.gz"

# --- Error Handling ---
if [ ! -d "$SOURCE_DIR" ]; then
   echo "ERROR: Source directory $SOURCE_DIR does not exist."
   echo "Backup failed."
   exit 1 # Exit with a failure code
fi

# --- Script Body ---
mkdir -p "$BACKUP_DIR"
echo "Starting backup of $SOURCE_DIR..."
tar -czvf "$BACKUP_DIR/$FILENAME" "$SOURCE_DIR"
echo "Backup complete! File created: $BACKUP_DIR/$FILENAME"
```

maintenance.sh:

```
#!/bin/bash
set -e    # Exit immediately if a command exits with a non-zero status.
echo "Starting system maintenance..."
# --- 1. Update Package Lists ---
# Fetches the list of available updates from the repositories.
echo "Updating package lists..."
sudo apt update
# --- 2. Perform Upgrades ---
# Installs the newest versions of all packages currently installed.
# The '-y' flag automatically answers 'yes' to all prompts.
echo "Installing system updates..."
sudo apt upgrade -y
# --- 3. Autoremove Old Packages ---
# Removes packages (like old kernels) that were automatically installed
# to satisfy dependencies but are no longer needed.
echo "Removing old and unused packages..."
sudo apt autoremove -y
# --- 4. Clean Package Cache ---
# Clears out the local cache of retrieved package files.
# This frees up disk space, as the .deb files are no longer needed
# After installation.
echo "Cleaning up package cache..."
sudo apt clean
echo "System maintenance complete!"
```

log_monitor.sh:

```
#!/bin/bash
set -e  # Exit immediately if a command fails
# --- Configuration ---
LOG_FILE="/var/log/auth.log"
KEYWORD="Failed"
# --- Error Handling ---
if [ ! -r "$LOG_FILE" ]; then
  echo "ERROR: Log file $LOG_FILE does not exist or is not readable."
  echo "Log monitor failed."
  exit 1 # Exit with a failure code
fi
# --- Script Body ---
echo "Starting log scan for '$KEYWORD' in $LOG_FILE..."
MATCHES=$(grep -i -n "$KEYWORD" "$LOG_FILE" || true)

if [ -n "$MATCHES" ]; then
  echo "------------------------------------"
  echo "ALERT: Potential issues found!"
  echo "------------------------------------"
  echo "$MATCHES"
else
  echo "Scan complete. No issues found."
fi
```

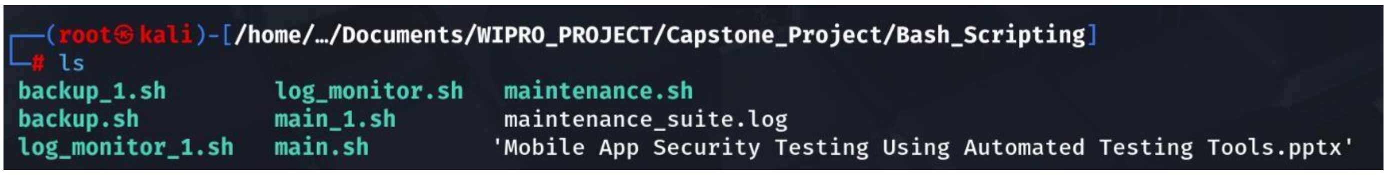
main.sh:

```
#!/bin/bash
# --- Configuration ---
BACKUP_SCRIPT="./backup_1.sh"
MAINTENANCE_SCRIPT="./maintenance.sh"
LOG_MONITOR_SCRIPT="./log_monitor_1.sh"
# 1. Define a central log file for the suite
LOG_FILE="maintenance_suite.log"
# Function to add a timestamped entry to the log
log_action() {
  echo "================ $(date) ================" >> "$LOG_FILE"
  echo "Running: $1" >> "$LOG_FILE"
  echo "-------------------------------------------- " >> "$LOG_FILE"
}
# --- Main Menu Loop ---
while true; do
  clear
  echo "======================================"
  echo "  System Maintenance Suite Menu"
  echo "======================================"
  echo "1. Run System Backup"
  echo "2. Run System Maintenance (Update & Clean)"
  echo "3. Monitor System Logs (Check for 'Failed')"
  echo "4. Exit"
  echo "------------------------------ "
  echo " (Logs are saved to $LOG_FILE)"
  echo -n "Enter your choice [1-4]: "
```

```
read choice
# --- Handle User Choice ---
case $choice in
  1)
    echo "Running backup script..."
    log_action "Backup Script"
    # Pipe all output (stdout & stderr) to 'tee'
    # -a : appends to the log file instead of overwriting
    $BACKUP_SCRIPT 2>&1 | tee -a "$LOG_FILE"
    ;;
  2)
    echo "Running maintenance script (requires sudo)..."
    log_action "Maintenance Script"
    sudo $MAINTENANCE_SCRIPT 2>&1 | tee -a "$LOG_FILE"
    ;;
  3)
    echo "Running log monitor (requires sudo)..."
    log_action "Log Monitor Script"
    sudo $LOG_MONITOR_SCRIPT 2>&1 | tee -a "$LOG_FILE"
    ;;
  4)
    echo "Exiting suite. Goodbye!"
    break
    ;;
  *)
    echo "Invalid choice. Please enter a number between 1 and 4."
    ;;
esac
```

```
    echo ""
    echo "Press Enter to return to the menu..."
    read
done
```

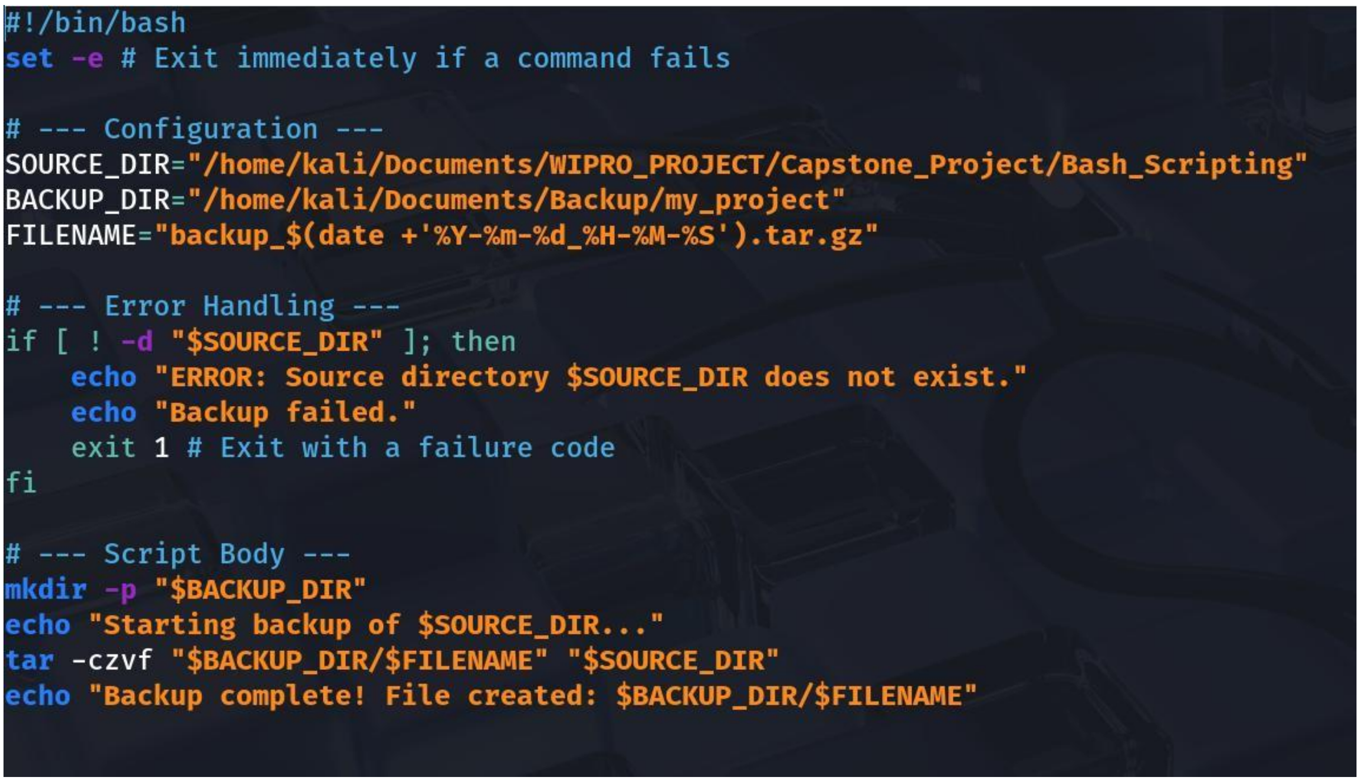
Screenshots:



```
┌──(root㉿kali)-[/home/…/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting]
└─# ls
 backup_1.sh        log_monitor.sh    maintenance.sh
 backup.sh          main_1.sh         maintenance_suite.log
 log_monitor_1.sh   main.sh          'Mobile App Security Testing Using Automated Testing Tools.pptx'
```

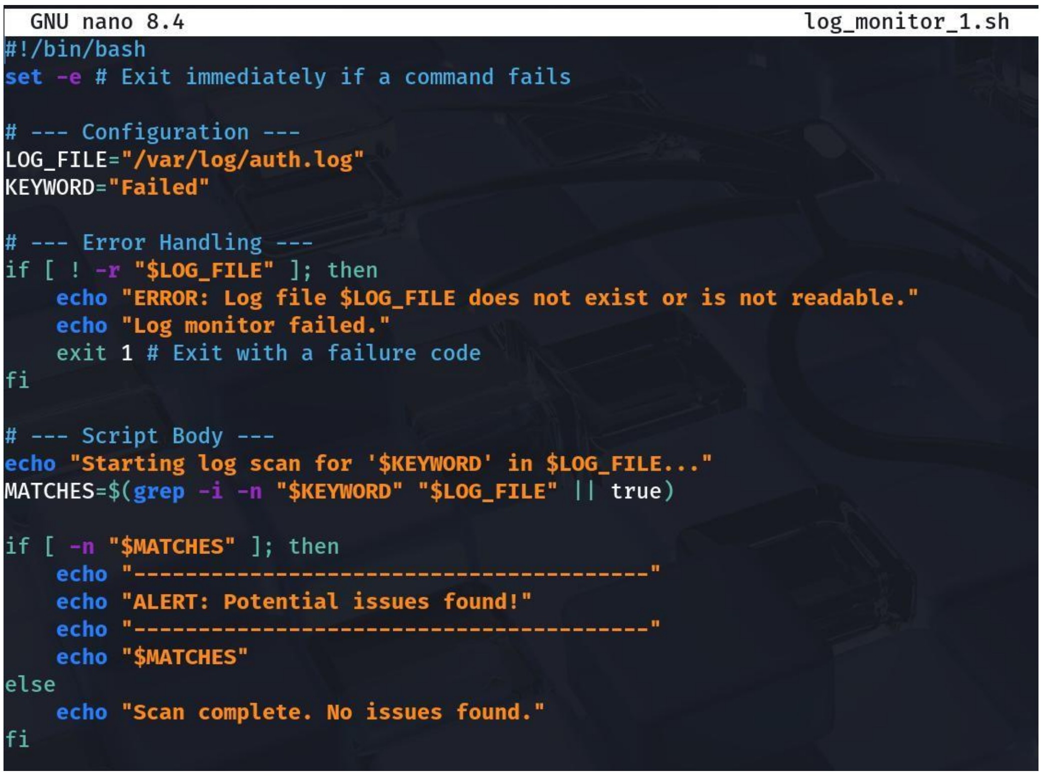
CODE:

backup.sh:



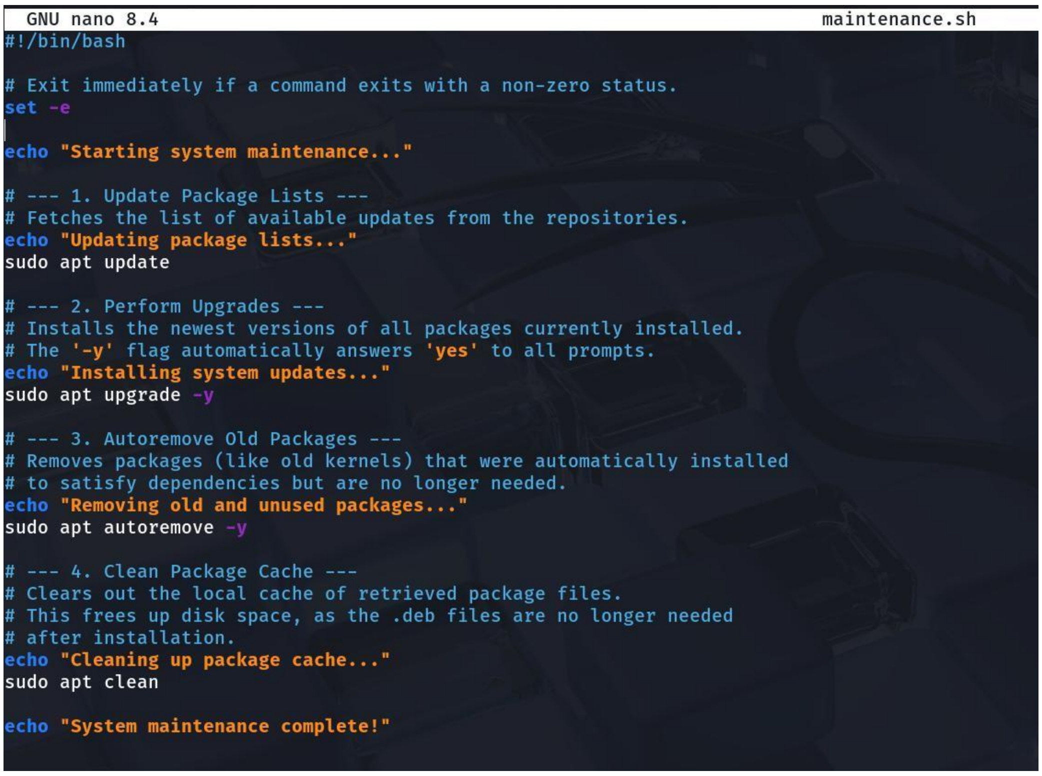
```
#!/bin/bash
set -e # Exit immediately if a command fails

# --- Configuration ---
SOURCE_DIR="/home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting"
BACKUP_DIR="/home/kali/Documents/Backup/my_project"
FILENAME="backup_$(date +'%Y-%m-%d_%H-%M-%S').tar.gz"

# --- Error Handling ---
if [ ! -d "$SOURCE_DIR" ]; then
    echo "ERROR: Source directory $SOURCE_DIR does not exist."
    echo "Backup failed."
    exit 1 # Exit with a failure code
fi

# --- Script Body ---
mkdir -p "$BACKUP_DIR"
echo "Starting backup of $SOURCE_DIR..."
tar -czvf "$BACKUP_DIR/$FILENAME" "$SOURCE_DIR"
echo "Backup complete! File created: $BACKUP_DIR/$FILENAME"
```

log_monitor.sh:



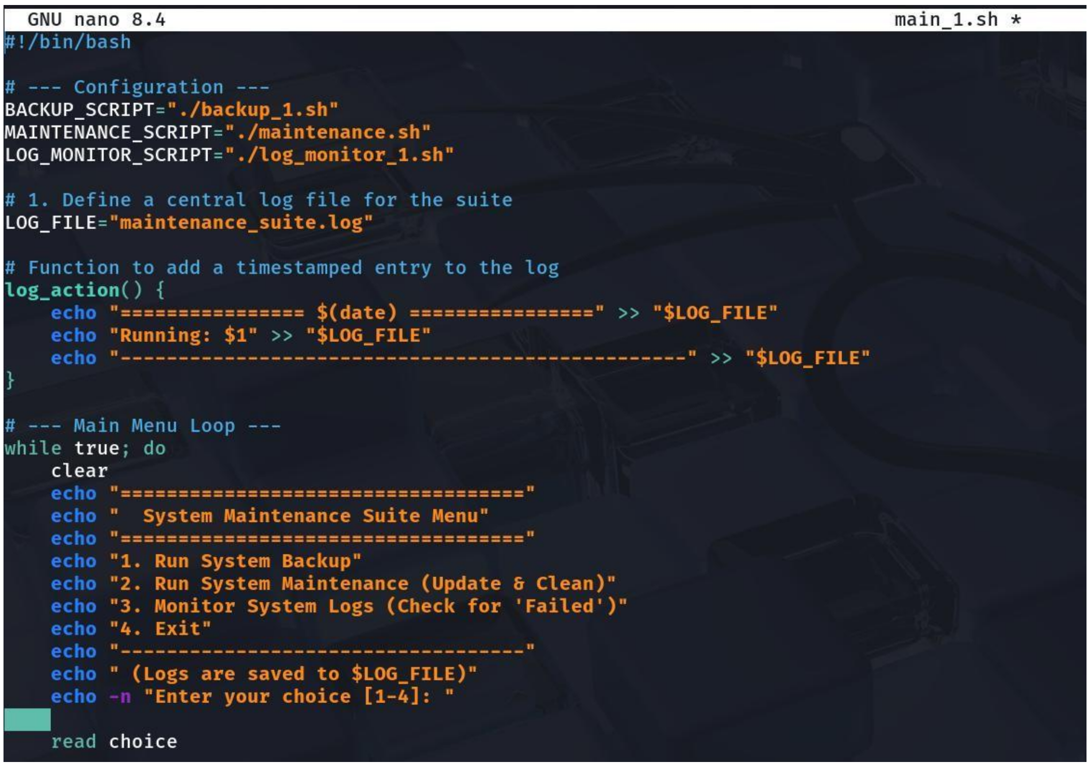
```
  GNU nano 8.4                                                 log_monitor_1.sh
#!/bin/bash
set -e # Exit immediately if a command fails

# --- Configuration ---
LOG_FILE="/var/log/auth.log"
KEYWORD="Failed"

# --- Error Handling ---
if [ ! -r "$LOG_FILE" ]; then
    echo "ERROR: Log file $LOG_FILE does not exist or is not readable."
    echo "Log monitor failed."
    exit 1 # Exit with a failure code
fi

# --- Script Body ---
echo "Starting log scan for '$KEYWORD' in $LOG_FILE..."
MATCHES=$(grep -i -n "$KEYWORD" "$LOG_FILE" || true)

if [ -n "$MATCHES" ]; then
    echo "-----------------------------------------"
    echo "ALERT: Potential issues found!"
    echo "-----------------------------------------"
    echo "$MATCHES"
else
    echo "Scan complete. No issues found."
fi
```

maintenance.sh:

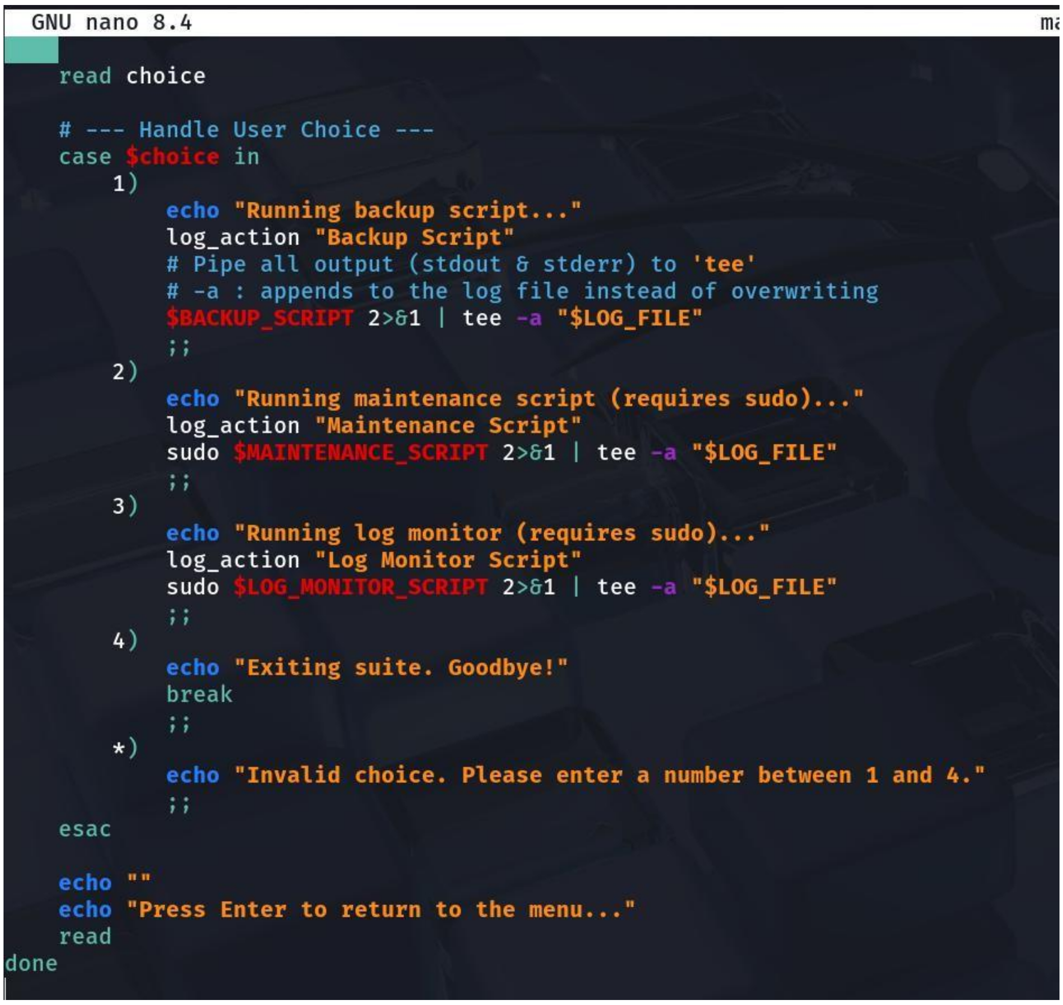


```
  GNU nano 8.4                                                     maintenance.sh
#!/bin/bash

# Exit immediately if a command exits with a non-zero status.
set -e

echo "Starting system maintenance..."

# --- 1. Update Package Lists ---
# Fetches the list of available updates from the repositories.
echo "Updating package lists..."
sudo apt update

# --- 2. Perform Upgrades ---
# Installs the newest versions of all packages currently installed.
# The '-y' flag automatically answers 'yes' to all prompts.
echo "Installing system updates..."
sudo apt upgrade -y

# --- 3. Autoremove Old Packages ---
# Removes packages (like old kernels) that were automatically installed
# to satisfy dependencies but are no longer needed.
echo "Removing old and unused packages..."
sudo apt autoremove -y

# --- 4. Clean Package Cache ---
# Clears out the local cache of retrieved package files.
# This frees up disk space, as the .deb files are no longer needed
# after installation.
echo "Cleaning up package cache..."
sudo apt clean

echo "System maintenance complete!"
```

main.sh:



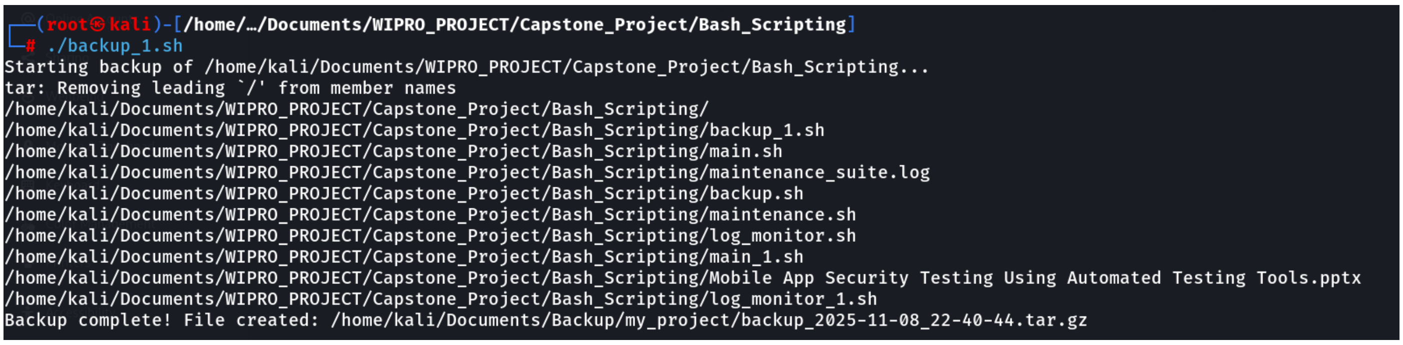
```
 GNU nano 8.4                                                  main_1.sh *
#!/bin/bash

# --- Configuration ---
BACKUP_SCRIPT="./backup_1.sh"
MAINTENANCE_SCRIPT="./maintenance.sh"
LOG_MONITOR_SCRIPT="./log_monitor_1.sh"

# 1. Define a central log file for the suite
LOG_FILE="maintenance_suite.log"

# Function to add a timestamped entry to the log
log_action() {
    echo "=============== $(date) ===============" >> "$LOG_FILE"
    echo "Running: $1" >> "$LOG_FILE"
    echo "-------------------------------------------------" >> "$LOG_FILE"
}

# --- Main Menu Loop ---
while true; do
    clear
    echo "===================================="
    echo "  System Maintenance Suite Menu"
    echo "===================================="
    echo "1. Run System Backup"
    echo "2. Run System Maintenance (Update & Clean)"
    echo "3. Monitor System Logs (Check for 'Failed')"
    echo "4. Exit"
    echo "------------------------------------"
    echo " (Logs are saved to $LOG_FILE)"
    echo -n "Enter your choice [1-4]: "

    read choice
```



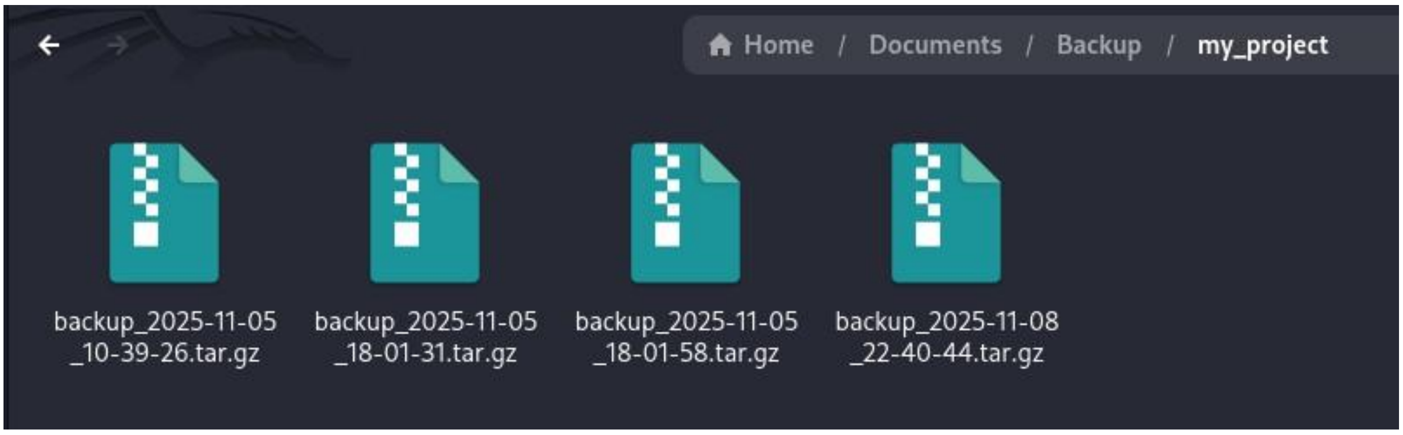
```
 GNU nano 8.4                                                          ma

    read choice

    # --- Handle User Choice ---
    case $choice in
        1)
            echo "Running backup script..."
            log_action "Backup Script"
            # Pipe all output (stdout & stderr) to 'tee'
            # -a : appends to the log file instead of overwriting
            $BACKUP_SCRIPT 2>&1 | tee -a "$LOG_FILE"
            ;;
        2)
            echo "Running maintenance script (requires sudo)..."
            log_action "Maintenance Script"
            sudo $MAINTENANCE_SCRIPT 2>&1 | tee -a "$LOG_FILE"
            ;;
        3)
            echo "Running log monitor (requires sudo)..."
            log_action "Log Monitor Script"
            sudo $LOG_MONITOR_SCRIPT 2>&1 | tee -a "$LOG_FILE"
            ;;
        4)
            echo "Exiting suite. Goodbye!"
            break
            ;;
        *)
            echo "Invalid choice. Please enter a number between 1 and 4."
            ;;
    esac

    echo ""
    echo "Press Enter to return to the menu..."
    read
done
```

OUTPUT:

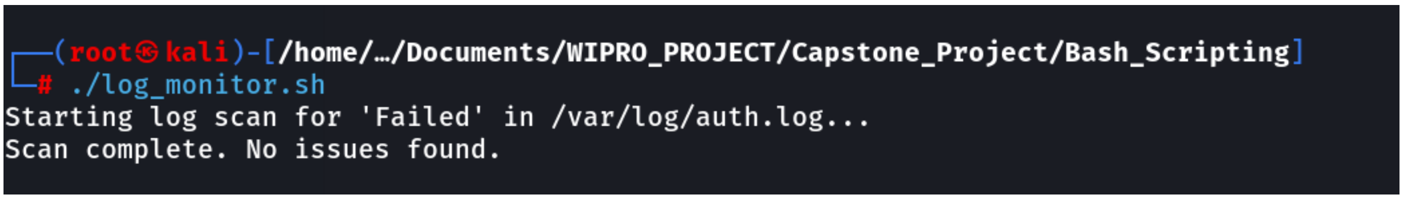
backup.sh:



```
┌──(root㉿kali)-[/home/…/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting]
└─# ./backup_1.sh
Starting backup of /home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting...
tar: Removing leading `/' from member names
/home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting/
/home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting/backup_1.sh
/home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting/main.sh
/home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting/maintenance_suite.log
/home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting/backup.sh
/home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting/maintenance.sh
/home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting/log_monitor.sh
/home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting/main_1.sh
/home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting/Mobile App Security Testing Using Automated Testing Tools.pptx
/home/kali/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting/log_monitor_1.sh
Backup complete! File created: /home/kali/Documents/Backup/my_project/backup_2025-11-08_22-40-44.tar.gz
```

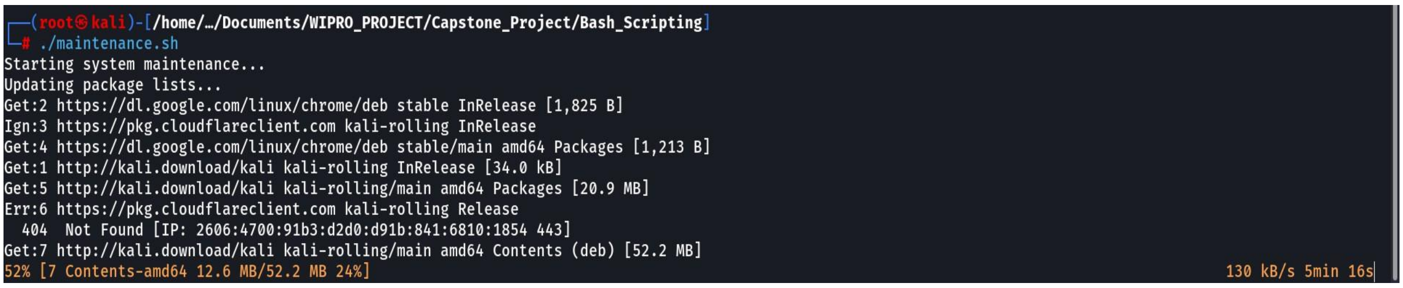

Home / Documents / Backup / my_project
backup_2025-11-05_10-39-26.tar.gz
backup_2025-11-05_18-01-31.tar.gz
backup_2025-11-05_18-01-58.tar.gz
backup_2025-11-08_22-40-44.tar.gz

log_monitor.sh:



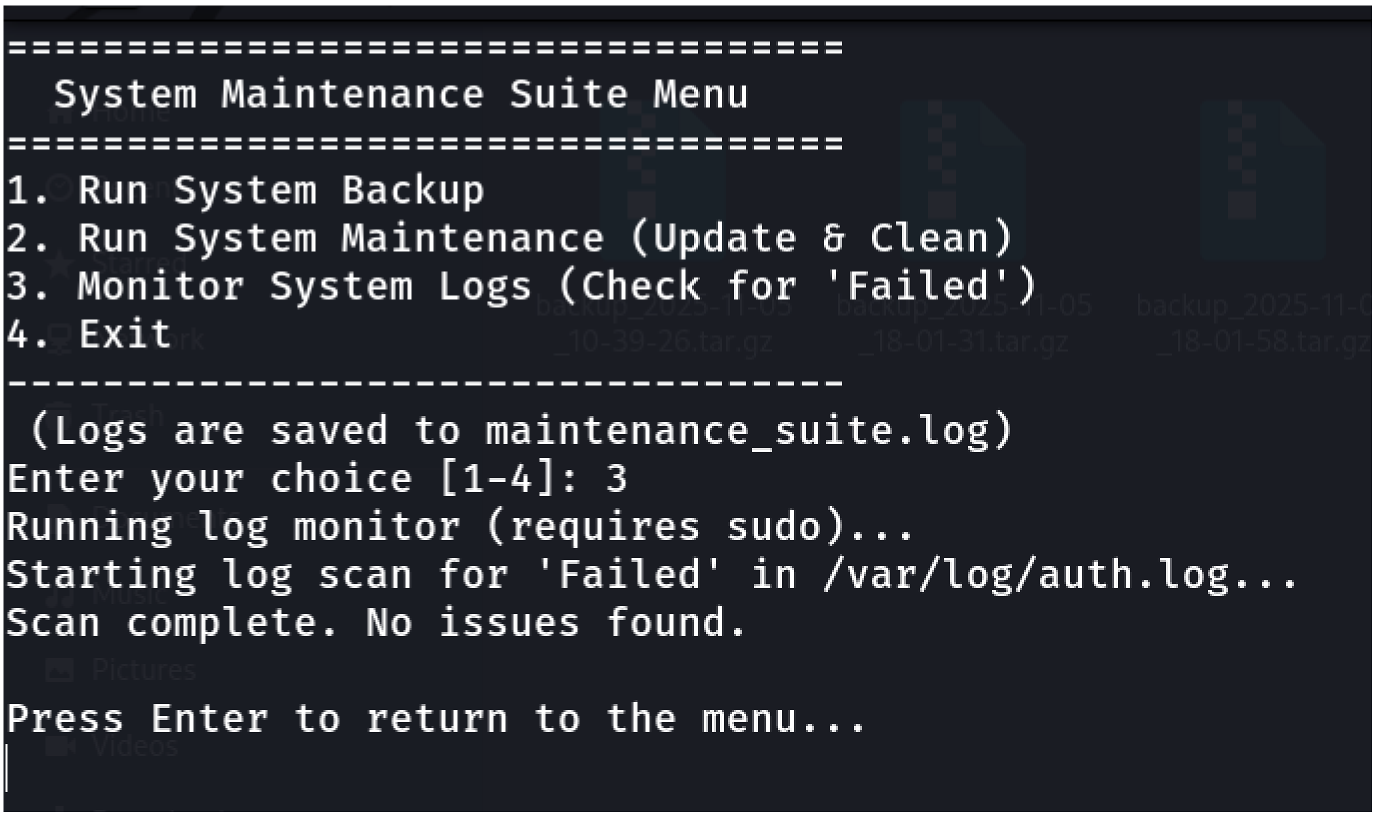
```
┌──(root㉿kali)-[/home/…/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting]
└─# ./log_monitor.sh
Starting log scan for 'Failed' in /var/log/auth.log...
Scan complete. No issues found.
```

maintenance.sh:



```
┌──(root㉿kali)-[/home/…/Documents/WIPRO_PROJECT/Capstone_Project/Bash_Scripting]
└─# ./maintenance.sh
Starting system maintenance...
Updating package lists...
Get:2 https://dl.google.com/linux/chrome/deb stable InRelease [1,825 B]
Ign:3 https://pkg.cloudflareclient.com kali-rolling InRelease
Get:4 https://dl.google.com/linux/chrome/deb stable/main amd64 Packages [1,213 B]
Get:1 http://kali.download/kali kali-rolling InRelease [34.0 kB]
Get:5 http://kali.download/kali kali-rolling/main amd64 Packages [20.9 MB]
Err:6 https://pkg.cloudflareclient.com kali-rolling Release
  404  Not Found [IP: 2606:4700:91b3:d2d0:d91b:841:6810:1854 443]
Get:7 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [52.2 MB]
52% [7 Contents-amd64 12.6 MB/52.2 MB 24%]                                              130 kB/s 5min 16s
```

main.sh:



```
=====================================
  System Maintenance Suite Menu
=====================================
1. Run System Backup
2. Run System Maintenance (Update & Clean)
3. Monitor System Logs (Check for 'Failed')
4. Exit
-------------------------------------
 (Logs are saved to maintenance_suite.log)
Enter your choice [1-4]: 3
Running log monitor (requires sudo)...
Starting log scan for 'Failed' in /var/log/auth.log...
Scan complete. No issues found.

Press Enter to return to the menu...
```

Conclusion:

This project successfully achieved its objective of creating a modular and functional Bash Scripting Suite for System Maintenance. By developing individual scripts for system backups, package updates, and log monitoring, and then integrating them into a single, menu-driven application, the suite effectively streamlines and simplifies common administrative workflows.

The implementation of centralized logging (to maintenance_suite.log) and basic error handling elevates the project from a simple collection of scripts to a robust utility. It provides a reliable and auditable tool that saves administrators time, reduces the potential for human error, and ensures that all maintenance actions are recorded. This project serves as a practical demonstration of how Bash scripting can be leveraged to build powerful, automated tools for real-world system management.