Capstone Project Report

1. Title

Bash Scripting Suite for System Maintenance

2. Objective

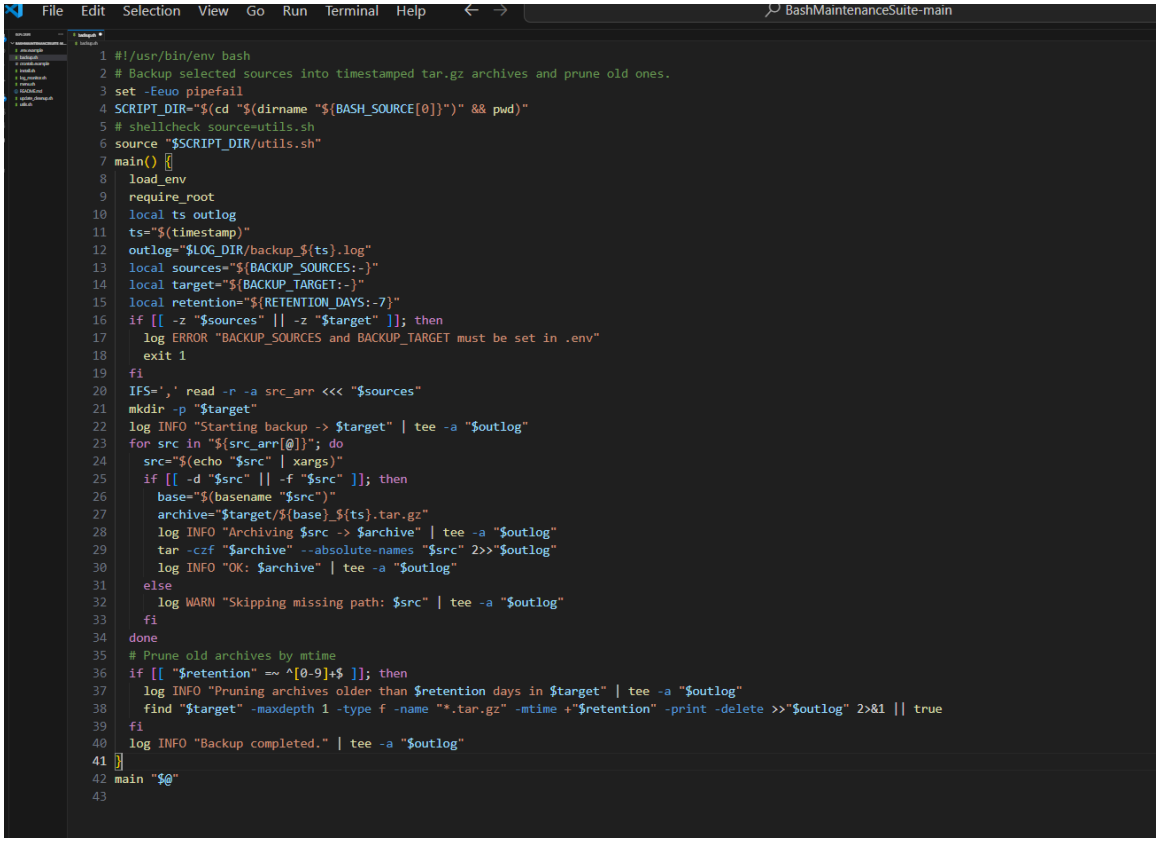
This project automates Linux system maintenance tasks such as backups, updates, cleanup, and log monitoring using Bash scripting. It is designed to simplify routine administrative work through automation and improve system reliability by scheduling essential maintenance tasks.

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

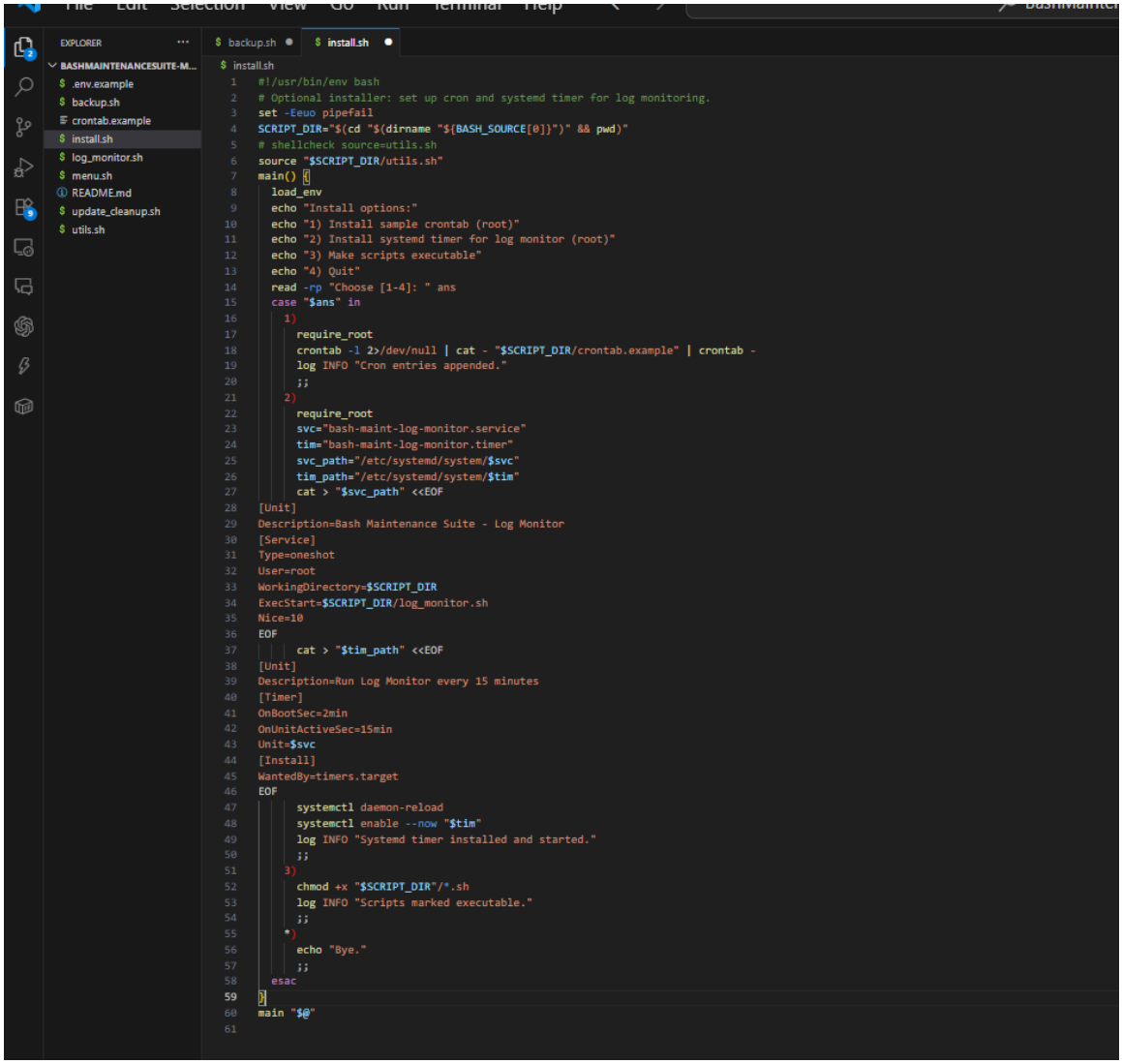
4. Codes

Below are the implemented Bash scripts used in the project:

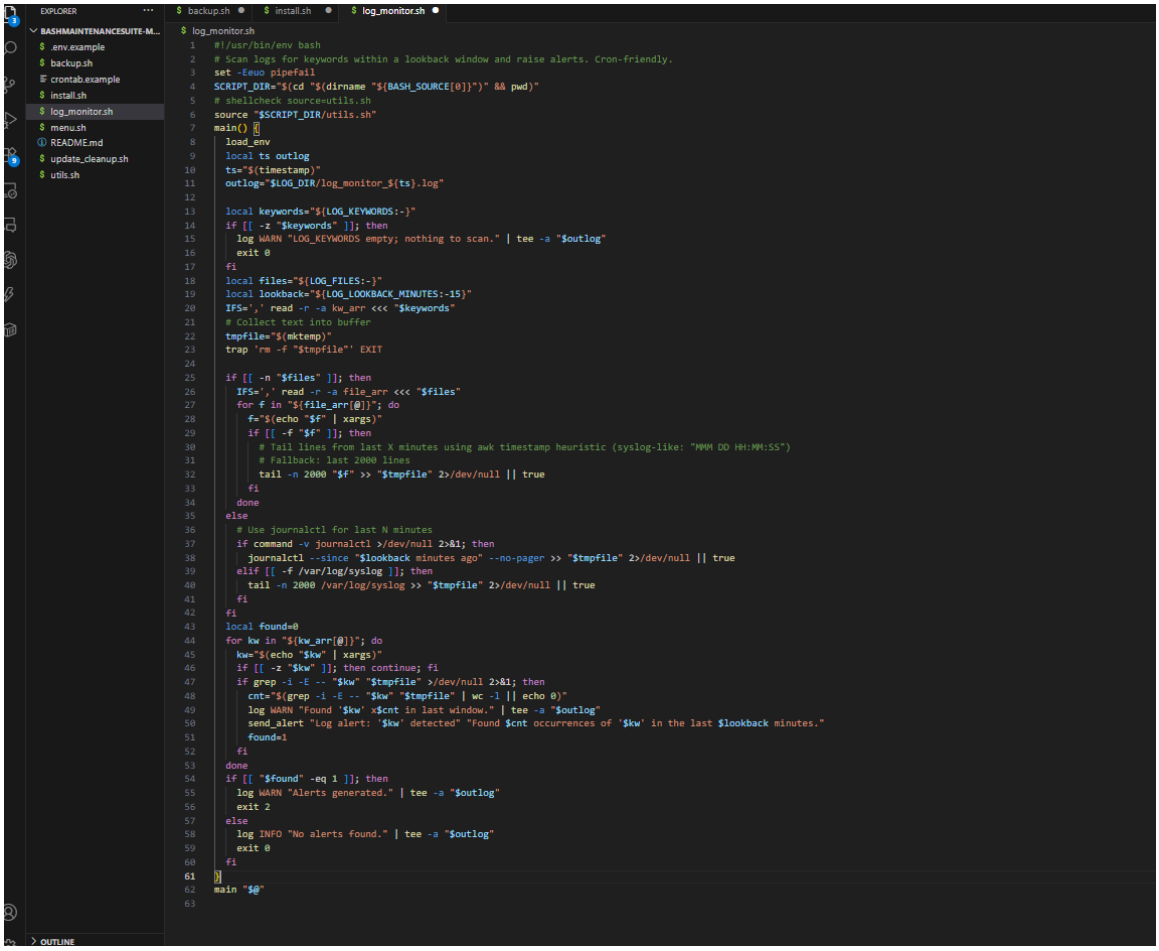



```bash
#!/usr/bin/env bash
# Backup selected sources into timestamped tar.gz archives and prune old ones.
set -Eeuo pipefail
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
# shellcheck source=utils.sh
source "$SCRIPT_DIR/utils.sh"
main() {
  load_env
  require_root
  local ts outlog
  ts="$(timestamp)"
  outlog="$LOG_DIR/backup_${ts}.log"
  local sources="${BACKUP_SOURCES:-}"
  local target="${BACKUP_TARGET:-}"
  local retention="${RETENTION_DAYS:-7}"
  if [[ -z "$sources" || -z "$target" ]]; then
    log ERROR "BACKUP_SOURCES and BACKUP_TARGET must be set in .env"
    exit 1
  fi
  IFS=',' read -r -a src_arr <<< "$sources"
  mkdir -p "$target"
  log INFO "Starting backup -> $target" | tee -a "$outlog"
  for src in "${src_arr[@]}"; do
    src="$(echo "$src" | xargs)"
    if [[ -d "$src" || -f "$src" ]]; then
      base="$(basename "$src")"
      archive="$target/${base}_${ts}.tar.gz"
      log INFO "Archiving $src -> $archive" | tee -a "$outlog"
      tar -czf "$archive" --absolute-names "$src" 2>>"$outlog"
      log INFO "OK: $archive" | tee -a "$outlog"
    else
      log WARN "Skipping missing path: $src" | tee -a "$outlog"
    fi
  done
  # Prune old archives by mtime
  if [[ "$retention" =~ ^[0-9]+$ ]]; then
    log INFO "Pruning archives older than $retention days in $target" | tee -a "$outlog"
    find "$target" -maxdepth 1 -type f -name "*.tar.gz" -mtime +"$retention" -print -delete >>"$outlog" 2>&1 || true
  fi
  log INFO "Backup completed." | tee -a "$outlog"
}
main "$@"
```

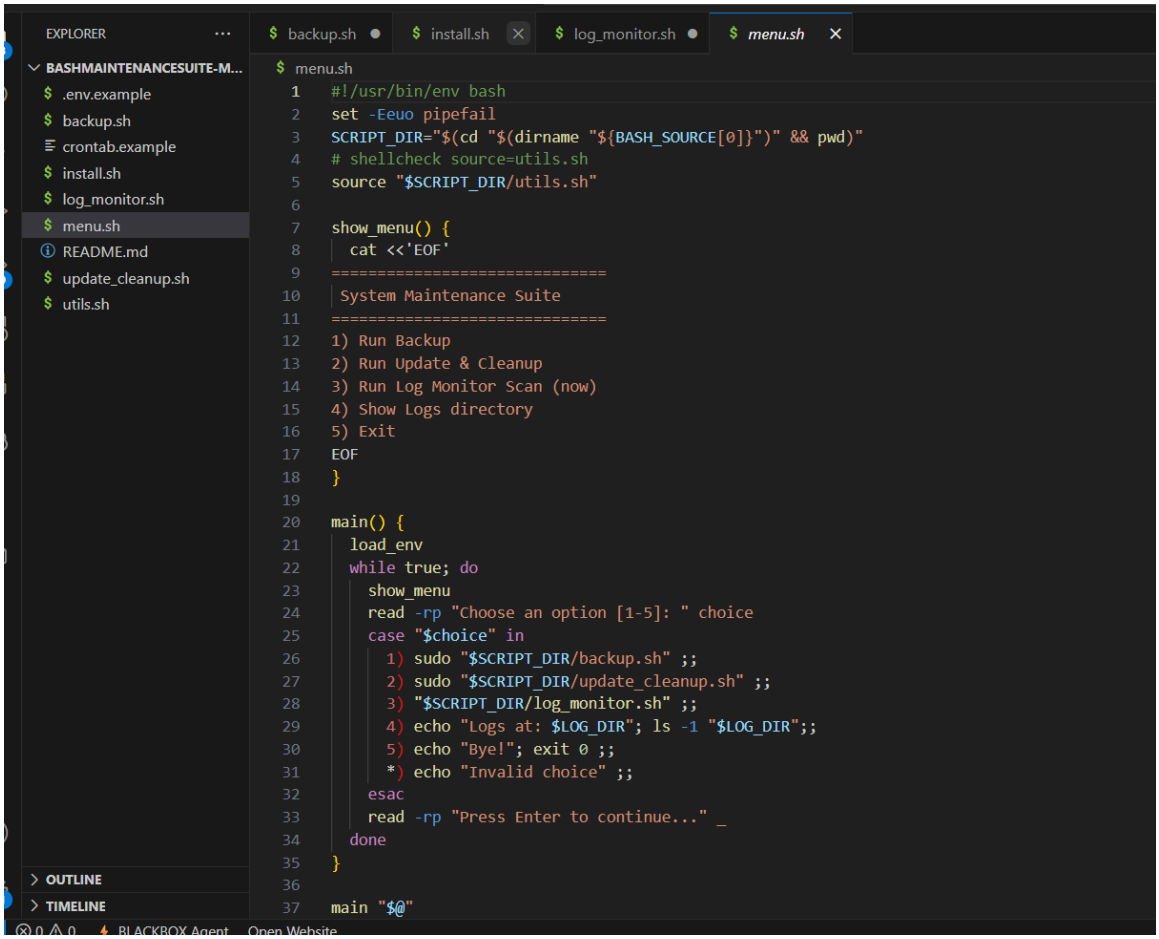
backup.sh


EXPLORER
BASHMAINTENANCESUITE-M...
.env.example
backup.sh
crontab.example
install.sh
log_monitor.sh
menu.sh
README.md
update_cleanup.sh
utils.sh
backup.sh
install.sh
install.sh
#!/usr/bin/env bash
Optional installer: set up cron and systemd timer for log monitoring.
set -Eeuo pipefail
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
# shellcheck source=utils.sh
source "$SCRIPT_DIR/utils.sh"
main() {
load_env
echo "Install options:"
echo "1) Install sample crontab (root)"
echo "2) Install systemd timer for log monitor (root)"
echo "3) Make scripts executable"
echo "4) Quit"
read -rp "Choose [1-4]: " ans
case "$ans" in
1)
require_root
crontab -l 2>/dev/null | cat - "$SCRIPT_DIR/crontab.example" | crontab -
log INFO "Cron entries appended."
;;
2)
require_root
svc="bash-maint-log-monitor.service"
tim="bash-maint-log-monitor.timer"
svc_path="/etc/systemd/system/$svc"
tim_path="/etc/systemd/system/$tim"
cat > "$svc_path" <<EOF
[Unit]
Description=Bash Maintenance Suite - Log Monitor
[Service]
Type=oneshot
User=root
WorkingDirectory=$SCRIPT_DIR
ExecStart=$SCRIPT_DIR/log_monitor.sh
Nice=10
EOF
cat > "$tim_path" <<EOF
[Unit]
Description=Run Log Monitor every 15 minutes
[Timer]
OnBootSec=2min
OnUnitActiveSec=15min
Unit=$svc
[Install]
WantedBy=timers.target
EOF
systemctl daemon-reload
systemctl enable --now "$tim"
log INFO "Systemd timer installed and started."
;;
3)
chmod +x "$SCRIPT_DIR"/*.sh
log INFO "Scripts marked executable."
;;
*)
echo "Bye."
;;
esac
}
main "$@"

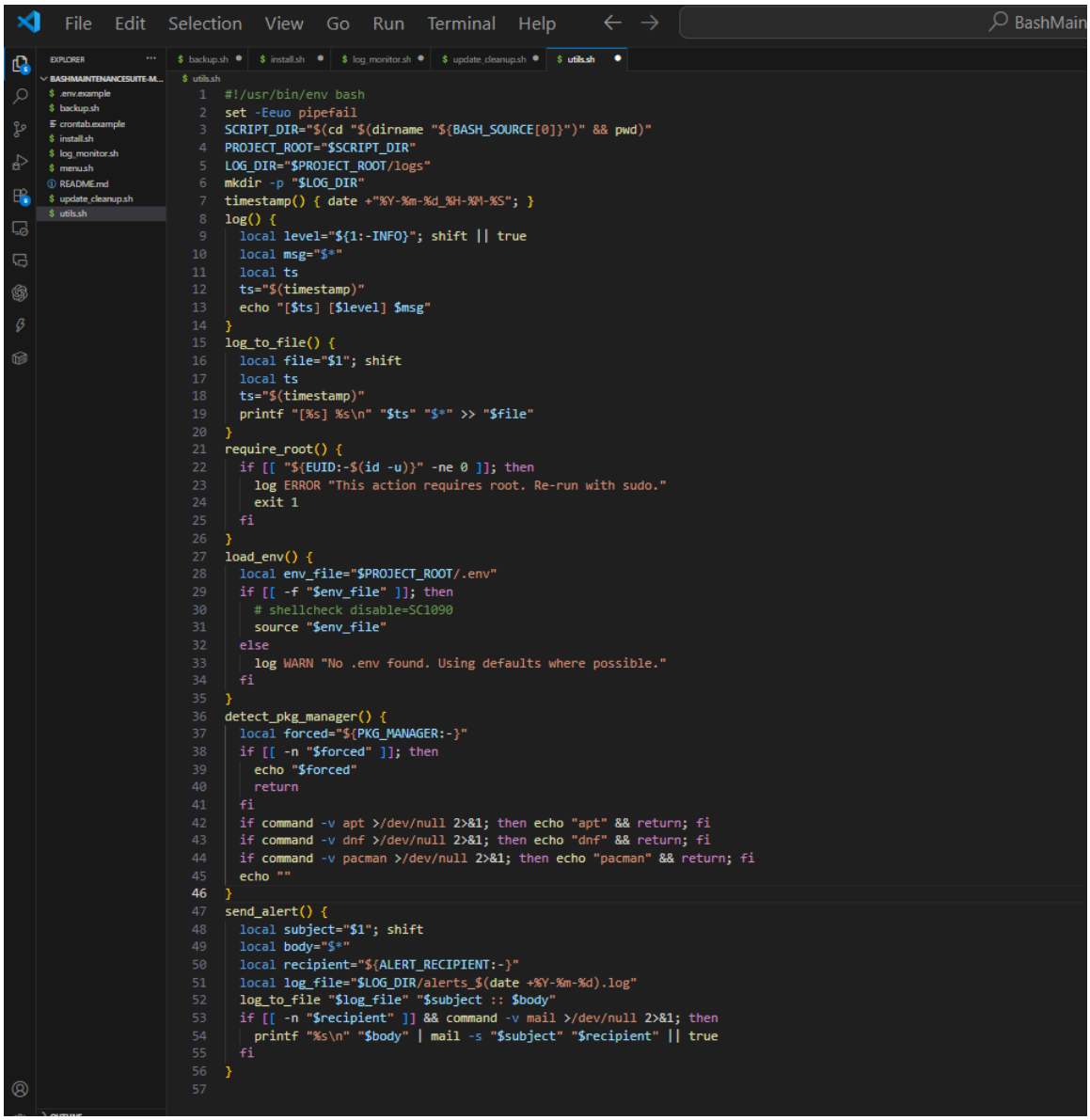
install.sh



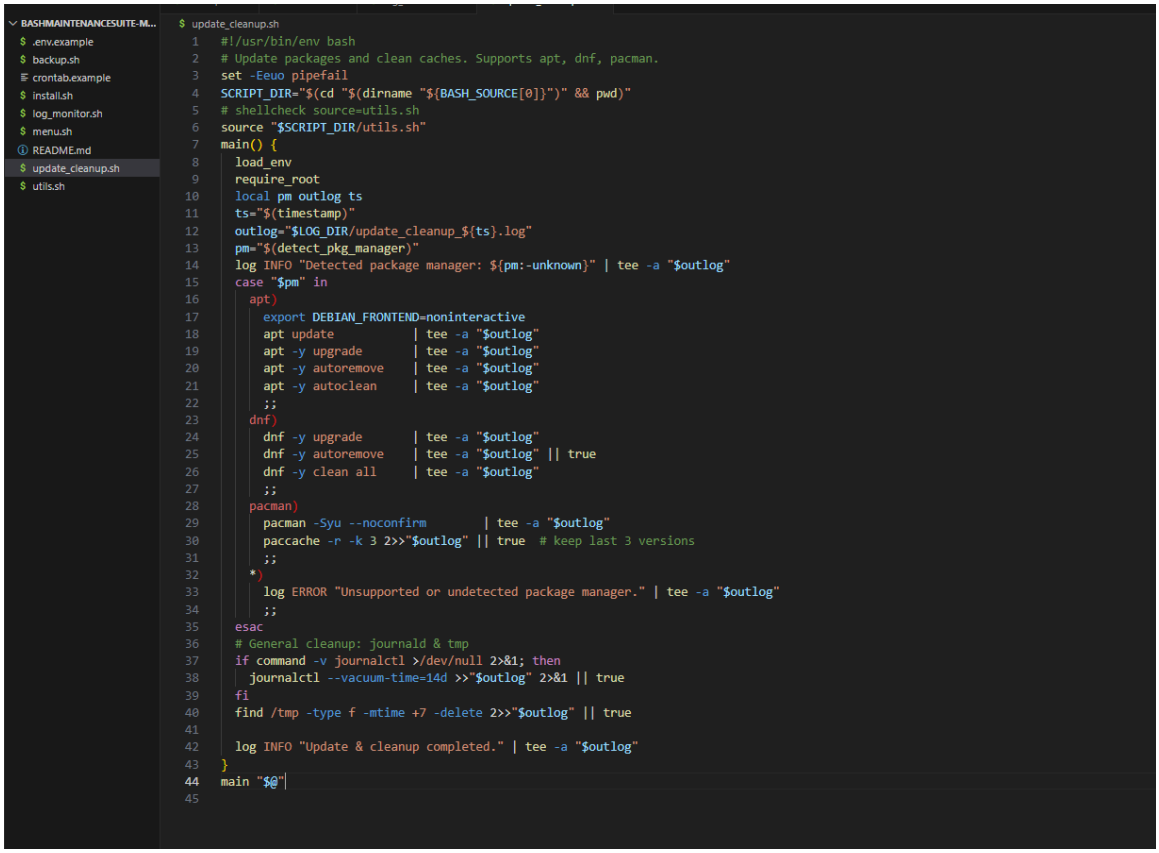
logmonitor.sh


EXPLORER
backup.sh
install.sh
log_monitor.sh
menu.sh
BASHMAINTENANCESUITE-M...
.env.example
backup.sh
crontab.example
install.sh
log_monitor.sh
menu.sh
README.md
update_cleanup.sh
utils.sh
OUTLINE
TIMELINE
#!/usr/bin/env bash
set -Eeuo pipefail
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
# shellcheck source=utils.sh
source "$SCRIPT_DIR/utils.sh"
show_menu() {
cat <<'EOF'
==============================
System Maintenance Suite
==============================
1) Run Backup
2) Run Update & Cleanup
3) Run Log Monitor Scan (now)
4) Show Logs directory
5) Exit
EOF
}
main() {
load_env
while true; do
show_menu
read -rp "Choose an option [1-5]: " choice
case "$choice" in
1) sudo "$SCRIPT_DIR/backup.sh" ;;
2) sudo "$SCRIPT_DIR/update_cleanup.sh" ;;
3) "$SCRIPT_DIR/log_monitor.sh" ;;
4) echo "Logs at: $LOG_DIR"; ls -1 "$LOG_DIR";;
5) echo "Bye!"; exit 0 ;;
*) echo "Invalid choice" ;;
esac
read -rp "Press Enter to continue..." _
done
}
main "$@"

menu.sh


File Edit Selection View Go Run Terminal Help
BashMain
EXPLORER
BASHMAINTENANCESUITE-M...
.env.example
backup.sh
crontab.example
install.sh
log_monitor.sh
menu.sh
README.md
update_cleanup.sh
utils.sh
backup.sh
install.sh
log_monitor.sh
update_cleanup.sh
utils.sh
utils.sh
#!/usr/bin/env bash
set -Eeuo pipefail
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
PROJECT_ROOT="$SCRIPT_DIR"
LOG_DIR="$PROJECT_ROOT/logs"
mkdir -p "$LOG_DIR"
timestamp() { date +"%Y-%m-%d_%H-%M-%S"; }
log() {
local level="${1:-INFO}"; shift || true
local msg="$*"
local ts
ts="$(timestamp)"
echo "[$ts] [$level] $msg"
}
log_to_file() {
local file="$1"; shift
local ts
ts="$(timestamp)"
printf "[%s] %s\n" "$ts" "$*" >> "$file"
}
require_root() {
if [["${EUID:-$(id -u)}" -ne 0]]; then
log ERROR "This action requires root. Re-run with sudo."
exit 1
fi
}
load_env() {
local env_file="$PROJECT_ROOT/.env"
if [[-f "$env_file"]]; then
# shellcheck disable=SC1090
source "$env_file"
else
log WARN "No .env found. Using defaults where possible."
fi
}
detect_pkg_manager() {
local forced="${PKG_MANAGER:-}"
if [[-n "$forced"]]; then
echo "$forced"
return
fi
if command -v apt >/dev/null 2>&1; then echo "apt" && return; fi
if command -v dnf >/dev/null 2>&1; then echo "dnf" && return; fi
if command -v pacman >/dev/null 2>&1; then echo "pacman" && return; fi
echo ""
}
send_alert() {
local subject="$1"; shift
local body="$*"
local recipient="${ALERT_RECIPIENT:-}"
local log_file="$LOG_DIR/alerts_$(date +%Y-%m-%d).log"
log_to_file "$log_file" "$subject :: $body"
if [[-n "$recipient"]] && command -v mail >/dev/null 2>&1; then
printf "%s\n" "$body" | mail -s "$subject" "$recipient" || true
fi
}

ulta.sh

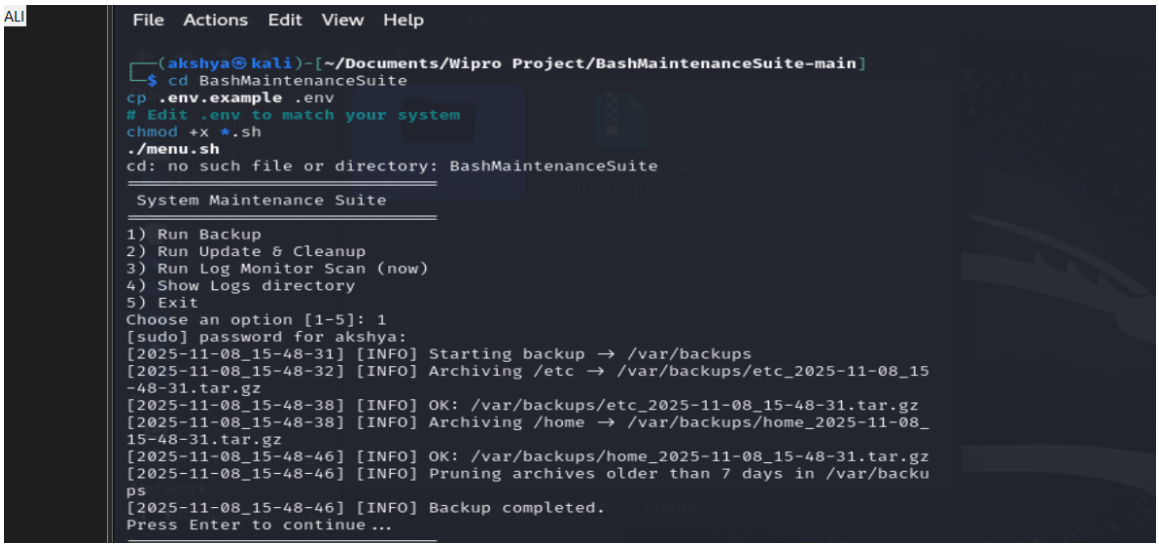


updatecleanup.sh

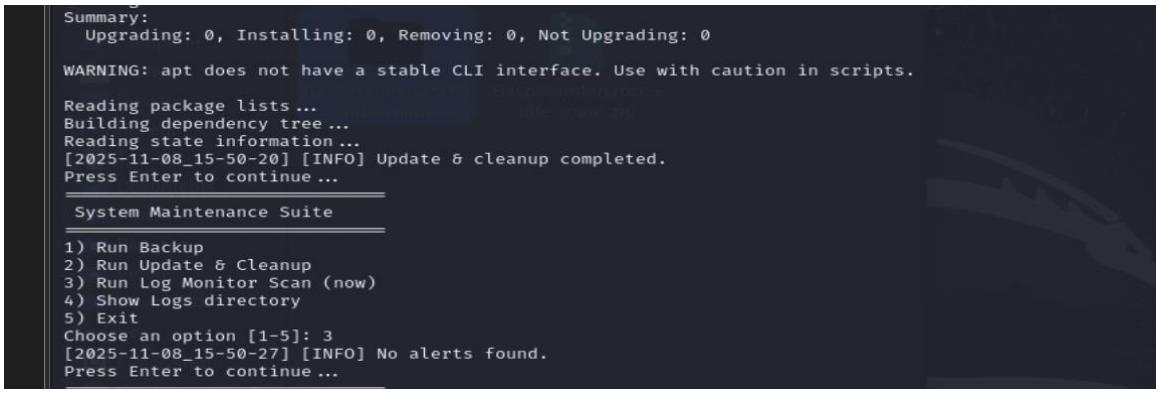
5. Full Execution Screenshots

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

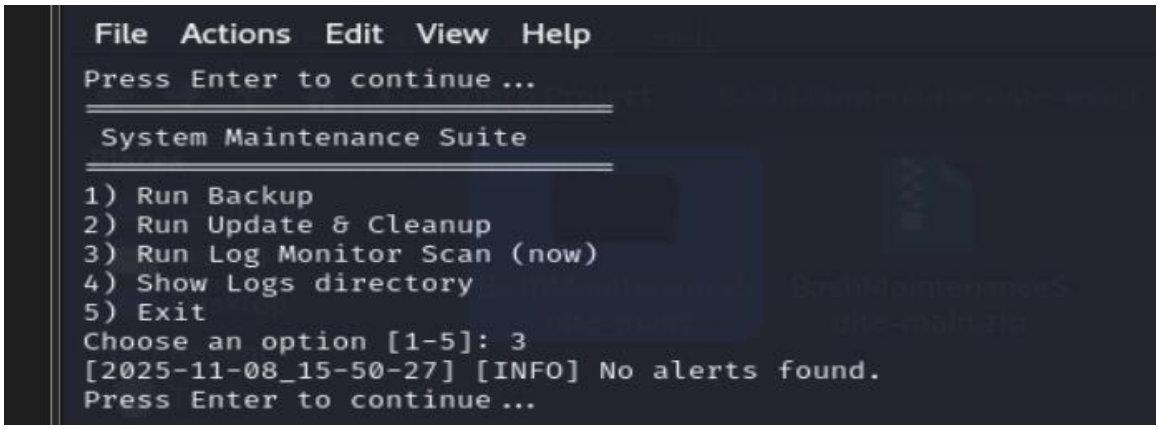
 Backup File Output



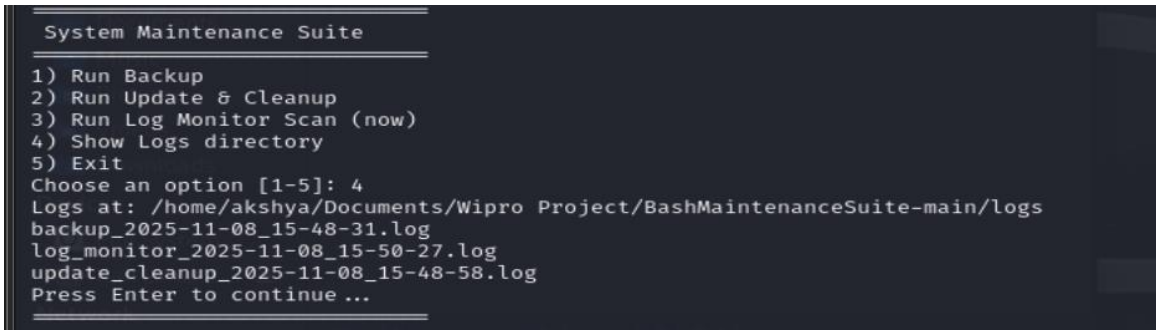
 System Update and Cleanup Output



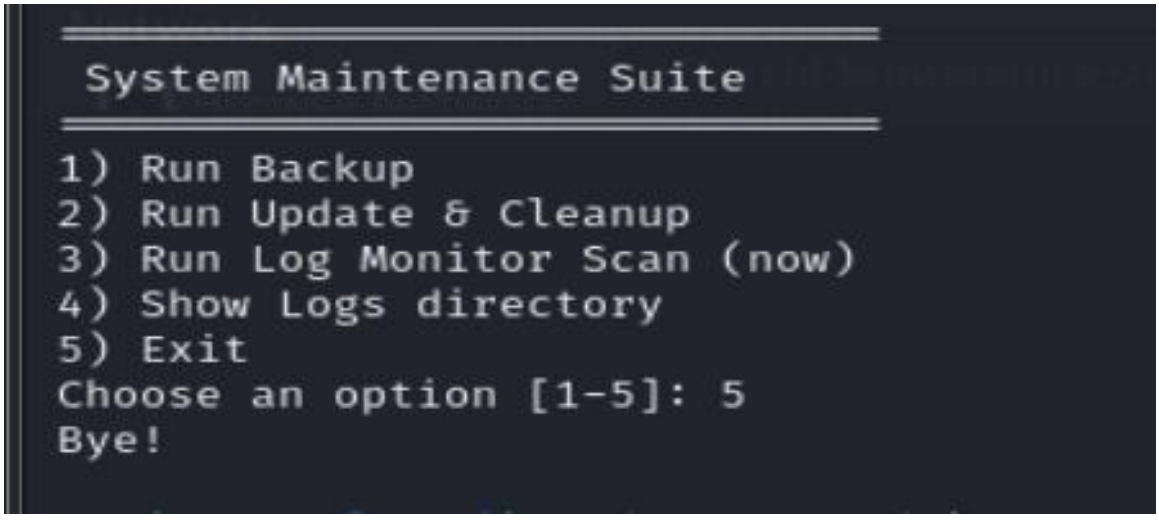
 Log Monitoring Output



 Show log directory



 Exit



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.