BASH SCRIPTING SUITE – SYSTEM AUTOMATION PROJECT

1. Introduction

The "Bash Scripting Suite – System Automation Project" is designed to automate essential system maintenance tasks using Bash scripts. In modern computing environments, system administrators and developers frequently perform repetitive tasks such as creating backups, updating the system, and monitoring logs. Automating these processes not only saves time but also reduces the chances of human error.

This project demonstrates the power of Bash scripting in simplifying administrative workflows, enhancing productivity, and ensuring system reliability. By integrating multiple scripts under a single menu-driven interface, users can easily perform maintenance tasks in an efficient and organized manner.

2. Objective

• Develop a set of Bash scripts that automate key system maintenance functions.
• Simplify user interaction through a menu-based interface for easy operation.
• Create an automated backup mechanism that archives selected directories securely.
• Implement a system update and cleanup routine that keeps the system optimized.
• Monitor logs effectively for error detection and system analysis.
• Demonstrate practical knowledge of Linux shell scripting and automation techniques applicable to real-world system administration tasks.

3. Methodology

The project is implemented using Bash scripting in a Linux-based (or WSL-enabled Windows) environment. The suite consists of multiple scripts that work together to provide a complete automation solution.

Step 1 – Design & Planning

• Identified key system maintenance operations to automate.
• Designed a modular approach with separate scripts for each functionality.

Step 2 – Implementation

1. menu.sh – Provides a simple terminal-based interface that allows users to select actions such as Backup, Update, Log Scan, or Exit.
2. backup.sh – Automates folder backup into a compressed .tar.gz archive and maintains a backup log.
3. update_clean.sh – Handles system update, upgrade, and cleanup operations for Linux environments.
4. log_watch.sh – Scans log files for errors or failed operations and reports them to the user.

Step 3 – Testing

• Tested all scripts in both Linux and Git Bash/WSL environments.
• Verified proper error handling, log creation, and menu functionality.

Step 4 – Integration

• Integrated all scripts into the menu.sh file, enabling centralized execution through a single menu interface.

2. CODES

menu.sh

```
#!/bin/bash
while true
do
 clear
 echo "=== SYSTEM MAINTENANCE MENU ==="
 echo "1. Backup Folder"
 echo "2. System Update & Cleanup"
 echo "3. Scan Logs"
 echo "4. Exit"
 echo "Enter choice:"
 read ch
 case $ch in
  1) bash backup.sh ;;
  2) bash update_clean.sh ;;
  3) bash log_watch.sh ;;
  4) exit ;;
  *) echo "Invalid option!" ;;
 esac
 echo "Press ENTER to continue..."
 read
done
```

backup.sh

```
cat > backup.sh << 'EOF'

#!/bin/bash

set -euo pipefail
```

```
now() { date '+%Y-%m-%d %H:%M:%S'; }
echo "Enter folder path to backup:"
read folder
if [ -z "${folder:-}" ] || [ ! -d "$folder" ]; then
  echo "[ERROR] $(now) Invalid folder: '$folder'"
  exit 1
fi
mkdir -p backups logs
base="$(basename "$folder")"
stamp="$(date +'%Y-%m-%d-%H-%M-%S')"
archive="backups/${base}-${stamp}.tar.gz"
proj="$(basename "$PWD")"

echo "----------------------------------------"
echo "[START]   $(now)"
echo "[SOURCE]  $folder"
echo "[ARCHIVE] $archive"
echo "----------------------------------------"
tar -czf "$archive" --exclude="$folder/$proj" "$folder"
echo "[DONE]    $(now)"
echo "[SAVED]   $archive"
echo "----------------------------------------"
{
  echo "$(now) [OK] $folder -> $archive"
} >> logs/backup.log
EOF
```

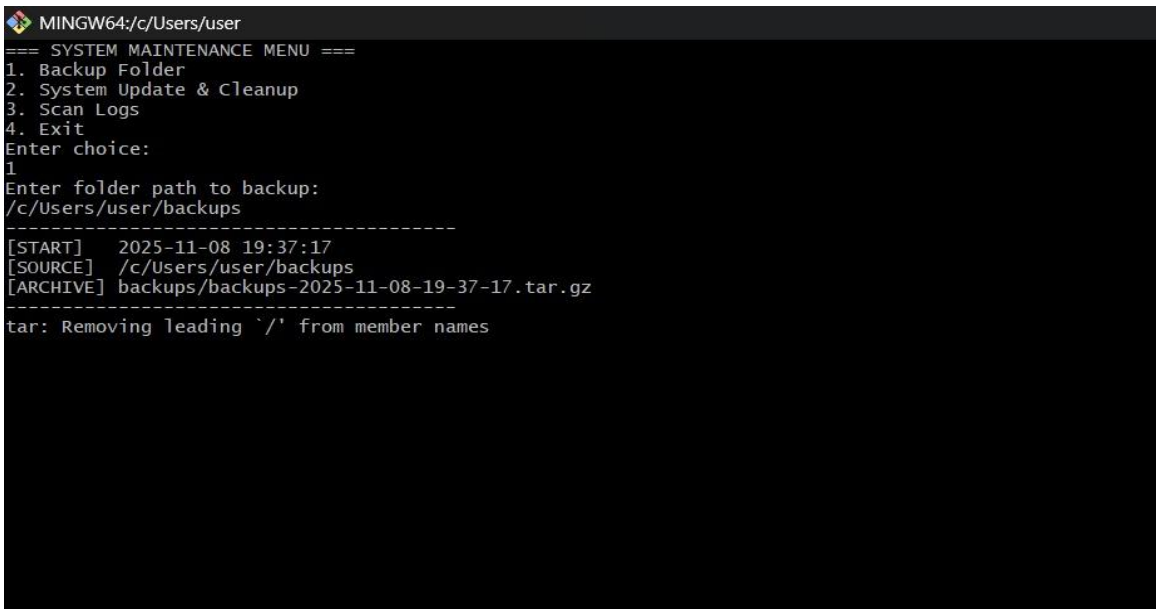
update_clean.sh

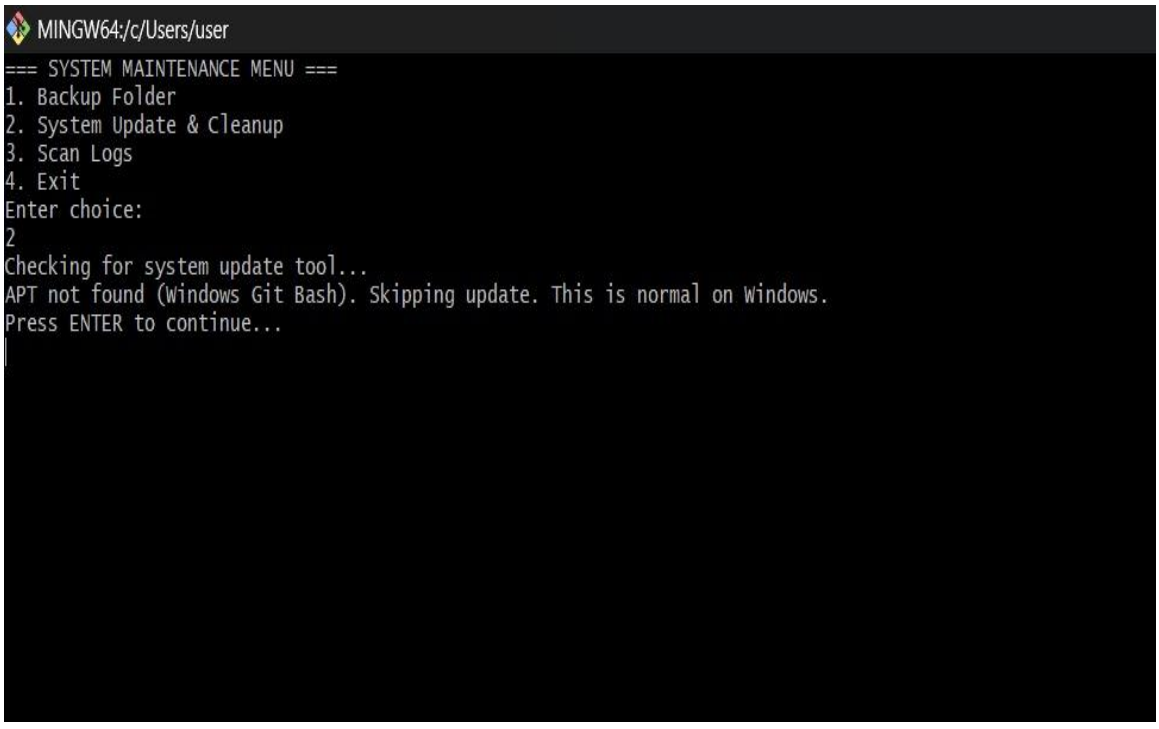
```
#!/bin/bash
echo "Checking for system update tool..."
if command -v apt >/dev/null 2>&1; then
  echo "APT found. Running system update..."
  sudo apt update
  sudo apt upgrade -y
  sudo apt autoremove -y
else
  echo "APT not found (Windows Git Bash). Skipping update. This is normal on Windows."
fi
```

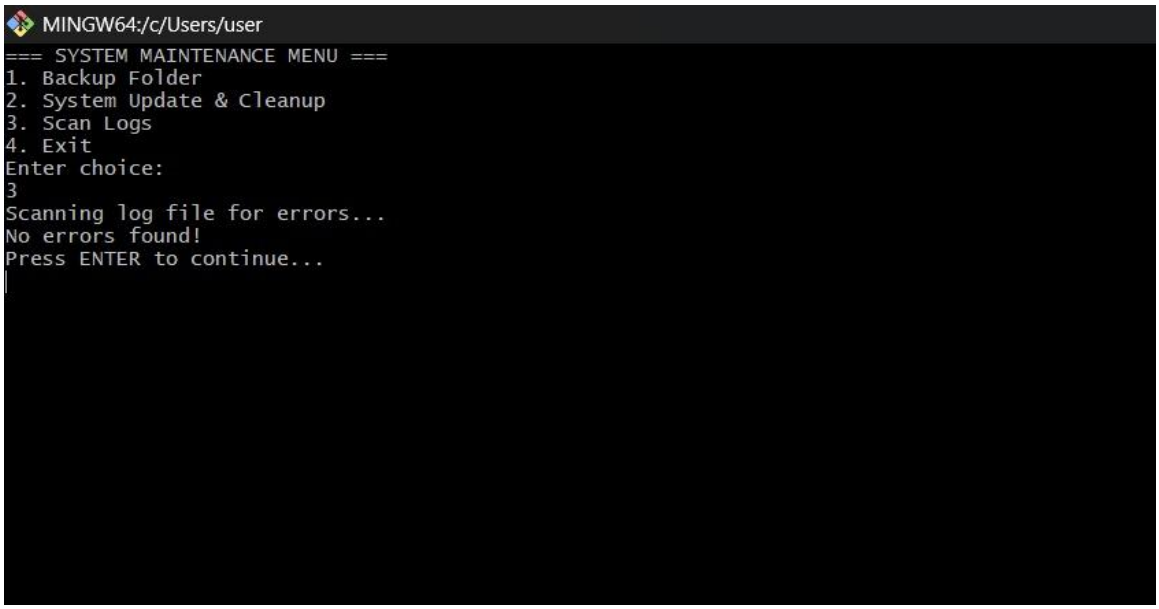
log_watch.sh

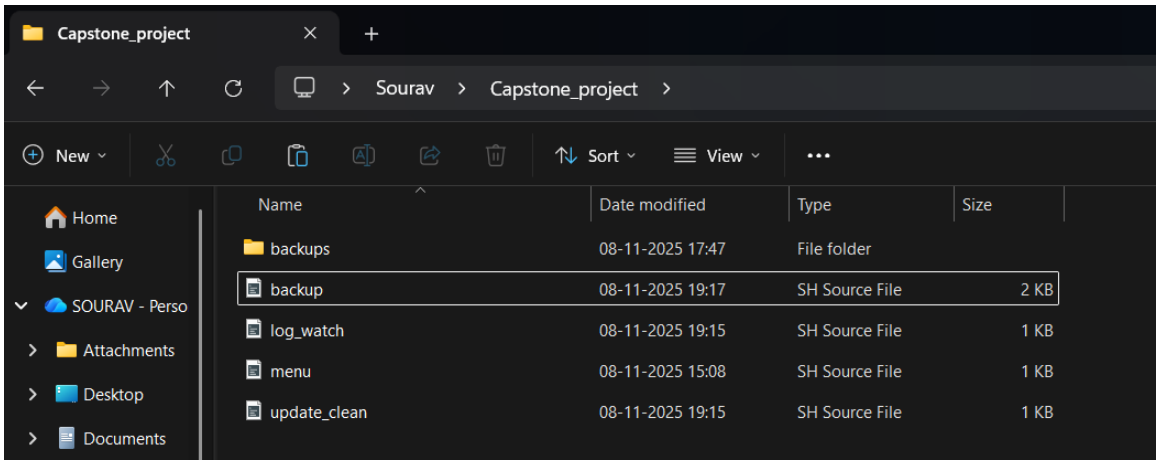
```
#!/bin/bash
log_file="update_clean.sh"
echo "Scanning log file for errors..."
if grep -Ei "ERROR|FAILED" "$log_file" >/dev/null 2>&1; then
  echo "Errors found:"
  grep -Ei "ERROR|FAILED" "$log_file"
else
  echo "No errors found!"
fi
```

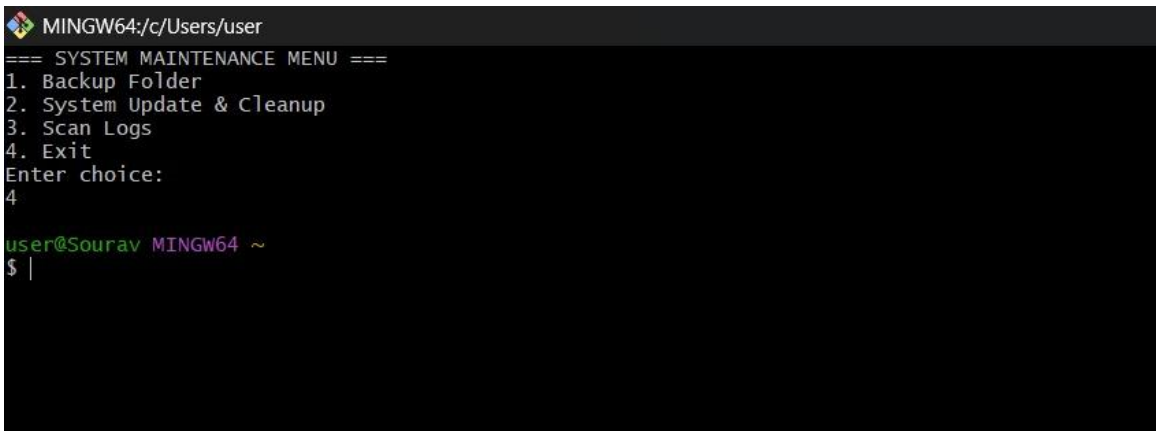
3. FULL SCREENSHOTS











4. GITHUB PUBLIC REPOSITORY LINK

https://github.com/SouravMO/CAPSTONE_PROJECT

7. Conclusion

The “Bash Scripting Suite – System Automation Project” successfully automates common maintenance operations, showcasing the effectiveness of Bash scripting in system management. Through this project, tasks such as file backups, system updates, and log scanning can be executed quickly and consistently with minimal user intervention.

The modular design allows easy customization and scalability for future system requirements. Overall, this project strengthens practical skills in Linux command-line operations, automation, and scripting — key competencies for system administrators and DevOps engineers. It serves as a valuable foundation for further advancements in automated system management and shell programming.