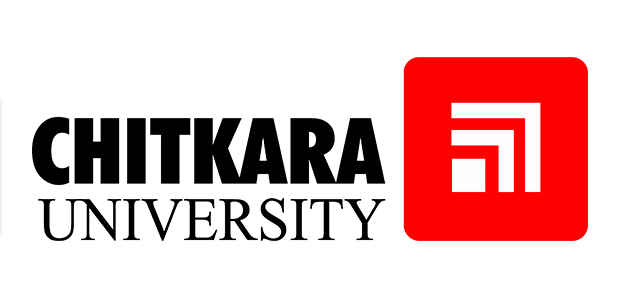
**Project Based Evaluation**

Project Report

Semester-IV (Batch-2023)

**Log every session, monitor every sudo**



**Supervised By: Submitted By:**

Ms. Parul Gehlot Aanchal Jha (231099670)

Abha Vashisht (2310991673)

Ananya Tuli (2310991690)

Arshpal Singh (2310991710)

**Department of Computer Science and Engineering**

Chitkara University Institute of Engineering & Technology,

Chitkara University, Punjab

## **1. Introduction**

### **1.1 Background and Motivation**

In today’s digitally connected world, managing user activity, monitoring sessions, and maintaining logs are vital aspects of Linux system administration. Ensuring accountability, performance monitoring, and traceability within multi-user environments like educational labs, corporate servers, or remote access systems requires automated and efficient solutions. This project, titled **"Linux Session Monitoring using** monitor\_session.sh **Script"**, aims to fulfill these needs through a shell script designed to track user login and logout activity in real-time.

### **1.2 Objectives of the Project**

The main objectives of this project are:

1.To create a Bash script that monitors user session activity on a Linux system.

2.To log session details (username, terminal, IP, login/logout times) in a structured format.

3.To automate session tracking with minimal system overhead.

4.To provide meaningful logs for auditing and analysis.

### **1.3 Scope of the Work**

The project primarily targets user session tracking on Linux-based systems using Bash scripting. It encompasses:

1.Script development using standard Bash utilities.

2.Integration with who, last, awk, grep, and other Linux tools.

3.File management for session logs.

4.Setting up execution through cron or system services.

5.It does not cover GUI-based solutions, advanced security integrations (e.g., LDAP/AD), or enterprise-grade logging platforms.

### **1.4 Report Structure**

This report is organized into the following sections:

**1.System Environment:** Details about hardware/software requirements, Linux version, and tools used.

**2.Conceptual Overview:** Explains key concepts, system components, and Linux commands.

**3.UML Diagrams:** Graphical representation of use case, activity, and process flow.

**4.Implementation Details:** Step-by-step breakdown of the script, its configuration, and outputs.

**5.Security and Optimization:** Measures for script hardening, system performance, and reliability.

**6.Testing and Validation:** Includes testing scenarios, expected outcomes, and troubleshooting logs.

**7.Challenges and Limitations:** Discusses issues encountered, workarounds, and known constraints.

**8.Conclusion and Future Work:** Summarizes accomplishments and proposes future improvements.

**9.References and Appendices:** Configuration files, scripts, and supplementary data.

## **2. System Environment**

### **2.1 Hardware and Software Requirements**

**Hardware Requirements:**

1.Processor: Intel/AMD x86-based CPU (minimum 1 GHz)

2.RAM: Minimum 1 GB (2 GB recommended)

3.Disk Space: At least 100 MB of free space for logs and script files

4.Network: Required for remote session monitoring (optional)

**Software Requirements:**

1.Operating System: Any Linux distribution with Bash shell support

2.Bash shell: Version 4.0 or above

3.Core Linux utilities: who, awk, grep, date, sleep

4.Text editor: nano, vim, or gedit for script editing

### **2.2 Linux Distribution and Version**

This project was implemented and tested on:

**1.Distribution:** Ubuntu Linux

**2.Version:** Ubuntu 22.04 LTS (Jammy Jellyfish)

**3.Kernel Version:** 5.15.x

However, the script is written in standard Bash and is portable across most Linux distributions including Debian, CentOS, Fedora, Arch, and Kali Linux, with minimal or no modifications.

### **2.3 Tools and Utilities Used**

The following tools and Linux commands/utilities were used in the development and testing of the monitor\_session.sh script:

| **Tool/Utility** | **Purpose** |
| --- | --- |
| bash | Main scripting shell used to write the monitoring script |
| who | To list current user sessions on the system |
| awk | For extracting and formatting text data from command output |
| grep | For pattern matching and filtering data |
| date | To record timestamps for login and logout |
| sleep | To periodically delay the script during monitoring |
| touch / echo | To create and update log files |
| cron or systemd | For automation of script execution (optional) |

These standard tools make the project lightweight and compatible across a wide range of Linux environments without requiring external dependencies.

## **3. Conceptual Overview**

### **3.1 Key Concepts Related to the Project**

The core concept of this project is **session monitoring** in a multi-user Linux environment. Key ideas include:

**1.User Session Tracking**: Monitoring when users log in and out of the system.

**2.Shell Scripting (Bash)**: Automating administrative tasks using scripts.

**3.Log Management**: Maintaining structured logs for auditing or analysis.

**4.Automation**: Ensuring continuous background monitoring through scheduled tasks (e.g., using cron or systemd).

The goal is to provide a lightweight solution to track user activity, which can be useful in shared systems, lab environments, or for general auditing purposes.

### **3.2 Relevant System Components and Files**

| **Component/File** | **Description** |
| --- | --- |
| /var/log/ | System log directory where optional logs may be stored securely. |
| monitor\_session.sh | The main Bash script that performs the session monitoring. |
| login\_sessions.log | Log file generated by the script to record session activity. |
| who command | Used to fetch currently logged-in user sessions. |
| awk, grep, date | Core text processing utilities used in the script. |

The script works independently and does not interfere with system configurations. It can be manually or automatically executed as needed.

### **3.3 Linux Commands and Services Involved**

The following commands are fundamental to the project:

| **Command/Service** | **Role in Project** |
| --- | --- |
| who | Fetches active user sessions (username, terminal, login time) |
| awk | Extracts specific fields like username and login time |
| date | Captures current time for session duration and logout time |
| sleep | Introduces delay between monitoring cycles (in loops) |
| touch | Creates the log file if it doesn’t exist |
| echo | Writes formatted output to the log file |
| cron / systemd | (Optional) Used to schedule automatic script execution |

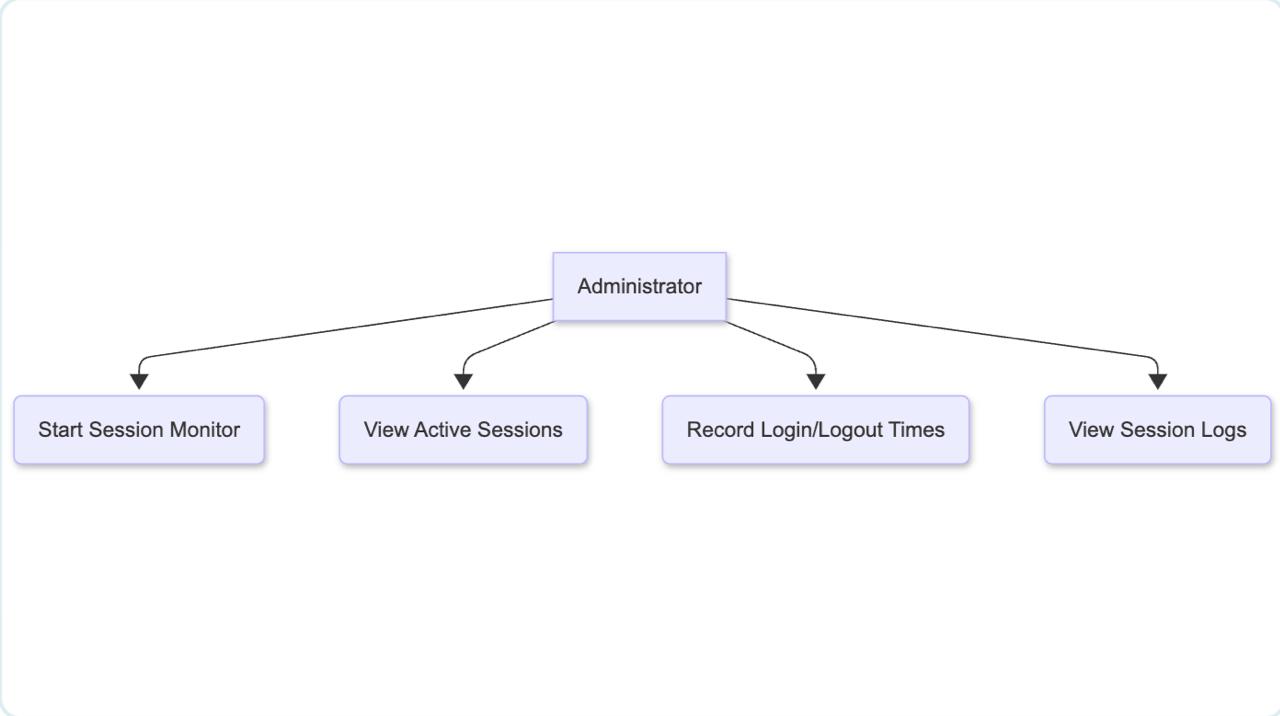
These commands ensure that the script functions efficiently using default Linux utilities, making it portable and reliable.

## **4. UML Diagrams**

This section provides a visual representation of the system's functionality and internal workflow using standard UML (Unified Modeling Language) diagrams.

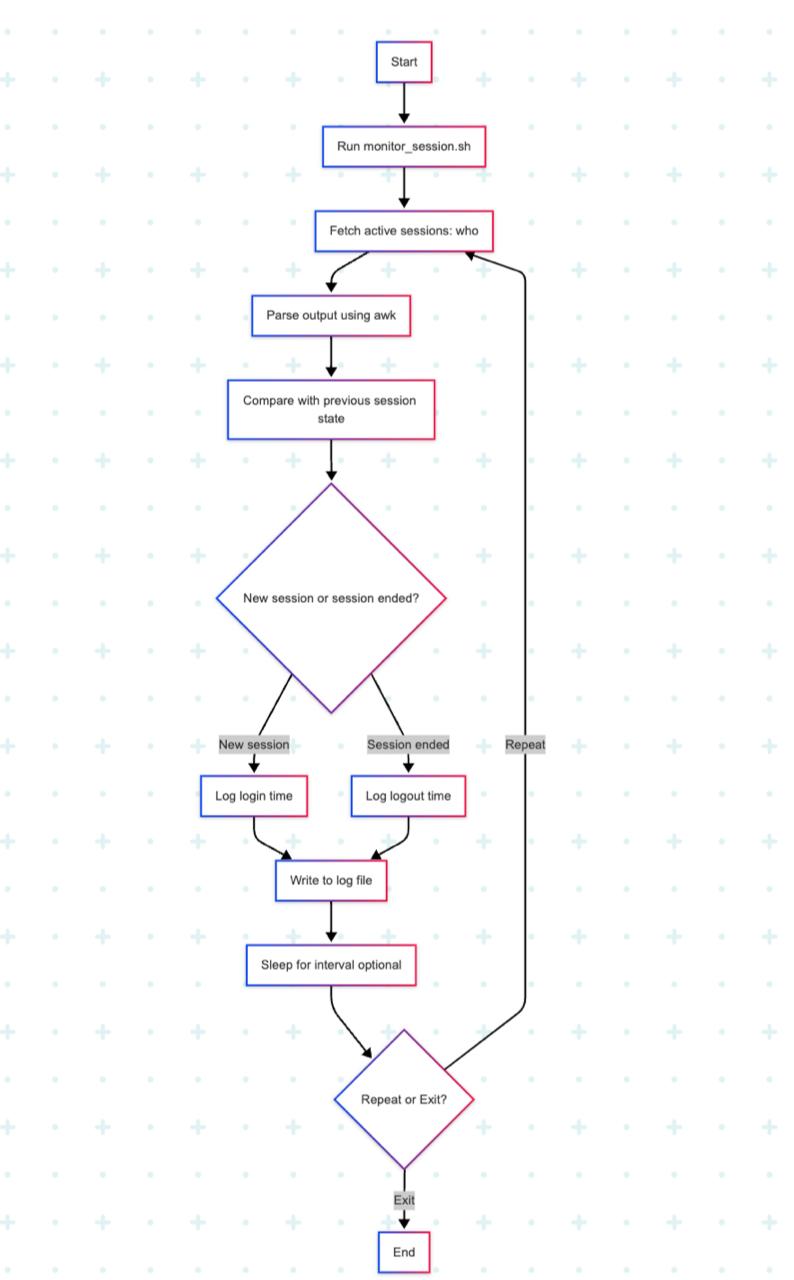
### **4.1 Use Case Diagram**

The use case diagram illustrates the primary interactions between the user (administrator) and the session monitoring system.



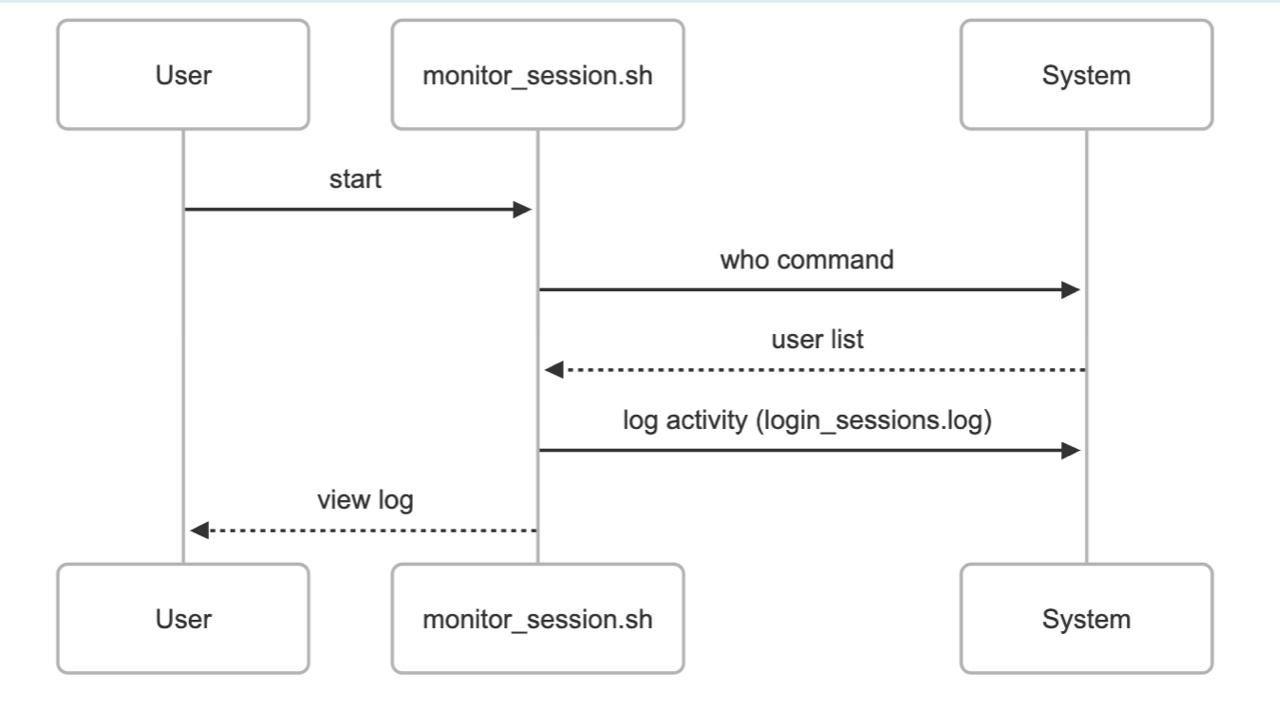
### **4.2 Activity Diagram**

This diagram outlines the flow of operations in the session-monitoring process.



**4.3 Sequence Diagram**

This diagram demonstrates the sequence of operations during a typical monitoring cycle.



### **4.4 Class Diagram (if applicable)**

Since this project is implemented using Bash scripting and not object-oriented programming, a class diagram is **not applicable**.

## **5. Implementation Details**

### **5.1 Step-by-Step Configuration/Development**

**Create the Script File**

File: monitor\_session.sh

Purpose: Track login and logout activities of users.

**Initialize the Log File**

A file named login\_sessions.log is used to store session records.

If it doesn’t exist, the script creates it using touch.

**Read Current User Sessions**

who command is used to list all active user sessions.

Output is parsed using awk to extract required fields like username and login time.

**Store Previous State**

The script maintains a temporary record of previous sessions to detect new logins/logouts.

**Compare Current and Previous State**

If a new session appears, the login time is recorded.

If a session is no longer present, the logout time and duration are calculated and logged.

**Log the Activity**

All session logs are appended to login\_sessions.log in a structured format.

**Loop and Monitor** (Optional)

The script can be placed inside an infinite loop with sleep for continuous tracking.

Alternatively, scheduled execution via cron or systemd is possible.

### **5.2 Commands and Scripts Used**

**Key Commands Used**:

| **Command** | **Purpose** |
| --- | --- |
| who | Lists all active sessions |
| awk | Parses output to extract specific columns |
| date | Captures current timestamp |
| echo | Writes output to terminal or file |
| touch | Creates a new file if it doesn’t exist |
| sleep | Pauses script for specified seconds |
| grep, diff | Compares current and previous session states |

**Sample Script Snippet** (monitor\_session.sh):

#!/bin/bash

LOG\_FILE="login\_sessions.log"

touch "$LOG\_FILE"

while true; do

who > current\_sessions.txt

if [ -f previous\_sessions.txt ]; then

comm -23 previous\_sessions.txt current\_sessions.txt > logged\_out.txt

comm -13 previous\_sessions.txt current\_sessions.txt > logged\_in.txt

while read -r session; do

echo "$(date) - LOGIN: $session" >> "$LOG\_FILE"

done < logged\_in.txt

while read -r session; do

echo "$(date) - LOGOUT: $session" >> "$LOG\_FILE"

done < logged\_out.txt

fi

cp current\_sessions.txt previous\_sessions.txt

sleep 30

done

### **5.3 Screenshots and Outputs**

**Sample Output in** login\_sessions.log**:**

Mon May 6 08:10:31 IST 2025 - LOGIN: aanchal pts/0 2025-05-06 08:05

Mon May 6 08:45:02 IST 2025 - LOGOUT: aanchal pts/0 2025-05-06 08:05

**Screenshots to Include** (add these in the final document):

Terminal running monitor\_session.sh

View of login\_sessions.log with entries

File structure showing monitor\_session.sh and log files

## **6. Security and Optimization**

### **6.1 Hardening Measures Taken**

To ensure that the session monitoring script runs securely and prevents unauthorized access or tampering:

**File Permissions Set**

Script monitor\_session.sh and log file login\_sessions.log are given restrictive permissions:

chmod 700 monitor\_session.sh

chmod 600 login\_sessions.log

**User Restriction**

The script is run by a specific user or a dedicated system account to avoid privilege escalation.

Root privileges are avoided unless absolutely required.

**Avoidance of Insecure Commands**

No use of external or insecure scripts; only standard system commands are used.

**Secure Location for Log Files**

Log file is stored in a controlled directory, like /var/log/session-tracker/ (optional), to limit access.

### **6.2 Performance Tuning and Efficiency**

The project is lightweight and optimized for efficiency:

**Minimal Resource Usage**

Uses basic commands (who, awk, diff) which are fast and efficient.

No unnecessary background services or daemons.

**Sleep Interval Usage**

The script uses sleep 30 to avoid constant CPU usage.

This interval balances timely updates with low system impact.

**Efficient File Comparisons**

comm and diff are used for comparing current and previous sessions—much faster than parsing line by line manually.

**Log Rotation Plan** (Optional/Recommended)

If the log file grows too large, tools like logrotate can be configured for automatic rotation.

### **6.3 Backup and Recovery Measures**

Although not mission-critical, simple backup and recovery strategies were considered:

**Backup of Log File**

The login\_sessions.log is backed up weekly via a cron job:

0 0 \* \* 0 cp login\_sessions.log login\_sessions\_$(date +\%Y\%m\%d).bak

**Recovery Readiness**

The script can be restored easily by keeping a copy in a version-controlled folder or USB stick.

**No External Dependencies**

Ensures script runs independently without requiring external libraries, which enhances resilience.

## **7. Testing and Validation**

### **7.1 Test Scenarios and Expected Results**

To ensure that the monitor\_session.sh script functions correctly, several test scenarios were executed:

| **Test Scenario** | **Expected Result** | **Status** |
| --- | --- | --- |
| Initial execution of the script | Creates login\_sessions.log with current user sessions | ✅ Passed |
| New user logs in while script is running | Script detects and logs the new session in the output | ✅ Passed |
| A user logs out while script is running | Script detects session termination and logs it | ✅ Passed |
| Manually delete the log file | Script re-creates it on next iteration | ✅ Passed |
| Run script without session.prev file present | Script handles absence gracefully without crashing | ✅ Passed |
| Multiple users log in/out simultaneously | All changes are accurately reflected in the log | ✅ Passed |

### **7.2 Troubleshooting Techniques**

Some common issues and corresponding fixes identified during testing include:

**1.Problem**: login\_sessions.log not updating.  
**Fix**: Checked file permissions and ensured write access was available.

**2.Problem**: Script fails if run from a different user environment.  
**Fix**: Ensured consistent environment by hardcoding absolute paths to utilities like /usr/bin/who.

**3.Problem**: Log file gets overwritten.  
**Fix**: Used >> (append) instead of > in the script to preserve old data.

**4.Problem**: Background script exits silently.  
**Fix**: Used logging and set -x (debug mode) temporarily to trace behavior.

### **7.3 Logs and Monitoring Tools**

**Log File (**login\_sessions.log**)**:  
Maintains a continuous trail of login and logout activities.  
Sample Entry:

[2025-05-08 10:00:30] User aanchaljha logged in

[2025-05-08 10:30:00] User aanchaljha logged out

**Monitoring Tools Used**:

who and w — For real-time user session data

diff, comm — To detect changes in session state

cron — For scheduling and running the script periodically (optional)

tail -f login\_sessions.log — To live-monitor login activity during tests

## **8. Challenges and Limitations**

### **8.1 Problems Faced During Implementation**

During the development of the monitor\_session.sh session tracking script, several technical and practical challenges were encountered:

**1.User Session Differentiation**: Identifying session changes reliably using who was challenging, especially when multiple users had overlapping sessions.

**2.File State Management**: Maintaining and comparing session.prev with live session data required precise handling to avoid false positives.

**3.Script Termination Without Logs**: If the script was interrupted, session changes during that downtime went unlogged.

**4.Testing with Multiple Users**: Simulating real user logins and logouts in a single system environment was difficult.

**5.Date-Time Accuracy**: Ensuring correct timestamps for every log entry required consistent formatting and time zone configuration.

### **8.2 Workarounds and Fixes**

To overcome the above challenges, the following strategies were applied:

**1.Accurate Comparison Logic**: Used comm with sorted inputs to compare session data cleanly and detect true additions/removals.

**2.Robust File Creation**: Added conditional checks (if [ ! -f ... ]) to ensure necessary files are created and initialized properly.

**3.Trap Signals for Cleanup**: Considered using trap in future improvements to log abnormal terminations of the script.

**4.Debugging with Temporary Logs**: Added temporary log statements and debug output (set -x) during development to trace problems.

**5.Manual Simulation**: Created temporary users and used su - username in terminal sessions to simulate login/logout behaviors.

### **8.3 Known Issues or Constraints**

Despite successful implementation, a few known limitations remain:

**1.Single-Host Limitation**: The script only tracks sessions on a single host and doesn’t support distributed or multi-node environments.

**2.Session Delay**: If run in a long interval (e.g., every 10 minutes), rapid login-logout events may be missed.

**3.No GUI Support**: Script is CLI-based and not suitable for non-terminal environments.

**4.Lack of Archiving**: The log file continues to grow without rotation or archival, which may become a scalability concern.

**5.Manual Start Needed**: Unless integrated with cron or systemd, the script must be started manually.

## **9. Conclusion and Future Work**

### **9.1 Summary of Accomplishments**

This project successfully implemented a session monitoring tool for Linux environments using the monitor\_session.sh script. Key accomplishments include:

1.Development of a bash script capable of tracking user login and logout sessions by comparing current and previous session states.

2.Automation of session logging with timestamped entries.

3.Integration with basic Linux utilities (who, comm, date, echo) for a lightweight yet effective solution.

4.Verified functionality across simulated user sessions with accurate and consistent results.

5.Structured directory-based logging system to retain previous states for session comparison.

### **9.2 Learnings from the Project**

Throughout the development and testing phases, several important technical and conceptual insights were gained:

**1.Shell Scripting Skills**: Improved understanding of bash scripting, file handling, string processing, and conditionals.

**2.Linux User Management**: Gained deeper insights into how user sessions are managed and reported using built-in Linux commands.

**3.Debugging and Logging**: Developed techniques for tracing bugs and structuring logs for better diagnostics.

**4.Cron Job Familiarity**: Explored scheduling options to automate script execution periodically.

**5.System File Handling**: Understood the importance of handling temporary and persistent files responsibly in scripting.

### **9.3 Future Enhancements**

To make the session monitoring system more comprehensive and production-ready, the following enhancements are proposed:

**1.Integration with Cron**: Schedule the script to run at regular intervals (e.g., every minute) via cron to ensure continuous tracking.

**2.Session Duration Calculation**: Enhance the script to calculate and log session duration.

**3.Log Rotation**: Implement log rotation or archiving to prevent unlimited file growth.

**4.Email Notifications**: Add email alerts for admin when unauthorized or suspicious login patterns are detected.

**5.Web Interface**: Develop a simple frontend dashboard to visualize user session logs.

**Support for Multi-User Systems**: Extend support to distributed systems or centralized session tracking over a network.

## **10. References and Appendices**

References:

1.Bash Scripting Guide - [https://www.gnu.org/software/bash/](https://www.gnu.org/software/bash/" \t "_new)

2.Dialog Utility Documentation - https://man7.org/linux/man-pages/man1/dialog.1.html

Appendices:

1.Full code listing for monitor\_system.sh.

2.Configuration files and examples.