CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

PROJECT REPORT ON USER AND GROUP MANAGEMENT

Program Name: BCA

Subject Name/Code: Linux Administration (23CAP-305)

Submitted by:

Name: Jyoti Kumari

UID: 23BCA10071

Section: BCA – 5’A

Submitted to:

Name: Ragini Chandel

Designation: Assistant-Professor

ABSTRACT

1. Introduction:

In Linux systems, user and group management forms a critical part of system administration. Managing users, assigning them to groups, and controlling access to directories ensures security, privacy, and efficient resource management.

This project implements a **menu-driven Bash script** that automates user and group management operations using sudo privileges. The script simplifies administrative tasks such as creating users and groups, managing permissions, and applying access control to directories, all through an interactive terminal menu.

2. Objectives:

The main objectives of this project are:

- To automate the process of user and group management using Bash scripting.
- To manage permissions and implement access control mechanisms.
- To provide an interactive menu-based interface for ease of use.
- To demonstrate the use of sudo for secure privilege escalation.

3. Techniques Used:

▪ **Shell Scripting Automation**: The project uses a Bash script to automate Linux administrative tasks related to user and group management, reducing manual effort and chances of error.

▪ **Menu-Driven Interface**: A terminal-based interactive menu guides users through various options such as creating users/groups, modifying permissions, and deleting accounts.

▪ **Sudo Privileges Enforcement**: All operations requiring elevated access are executed with sudo, ensuring only authorized users can perform administrative actions.

▪ **User & Group Management**: The script implements core Linux commands (useradd, groupadd, usermod, etc.) to create, delete, and modify users and groups.

▪ **Directory Permission Control**: The script allows setting or modifying access permissions and ownership (chown, chmod) on directories to manage user access levels effectively.

▪ **Error Handling & Validation**: Input validation and condition checking ensure that invalid or duplicate user/group entries are not processed, maintaining system integrity.

▪ **Role-Based Access Simulation**: By assigning users to specific groups and managing directory access, the script simulates basic role-based access control (RBAC).

▪ **Portable and Reusable**: The script can be reused on different Linux distributions with little to no modification, as it uses standard commands and syntax.

4. System Configuration:

4.1 Software Requirements

The software requirements for developing and executing the Linux User and Group Management project are as follows:

- Operating System: Linux (Ubuntu / CentOS / Rocky / Debian)
- Shell: Bash (default in most Linux distros)
- sudo package installed
- Terminal access with basic privileges

4.2 Hardware Requirements

The hardware specifications recommended for the smooth development and execution of the Linux User and Group Management Bash Script are as follows:

- Processor: Any x86/x64 CPU
- Memory: Minimum 1 GB RAM
- Storage: Minimum 2 GB free space

5. Tools and Commands Used:

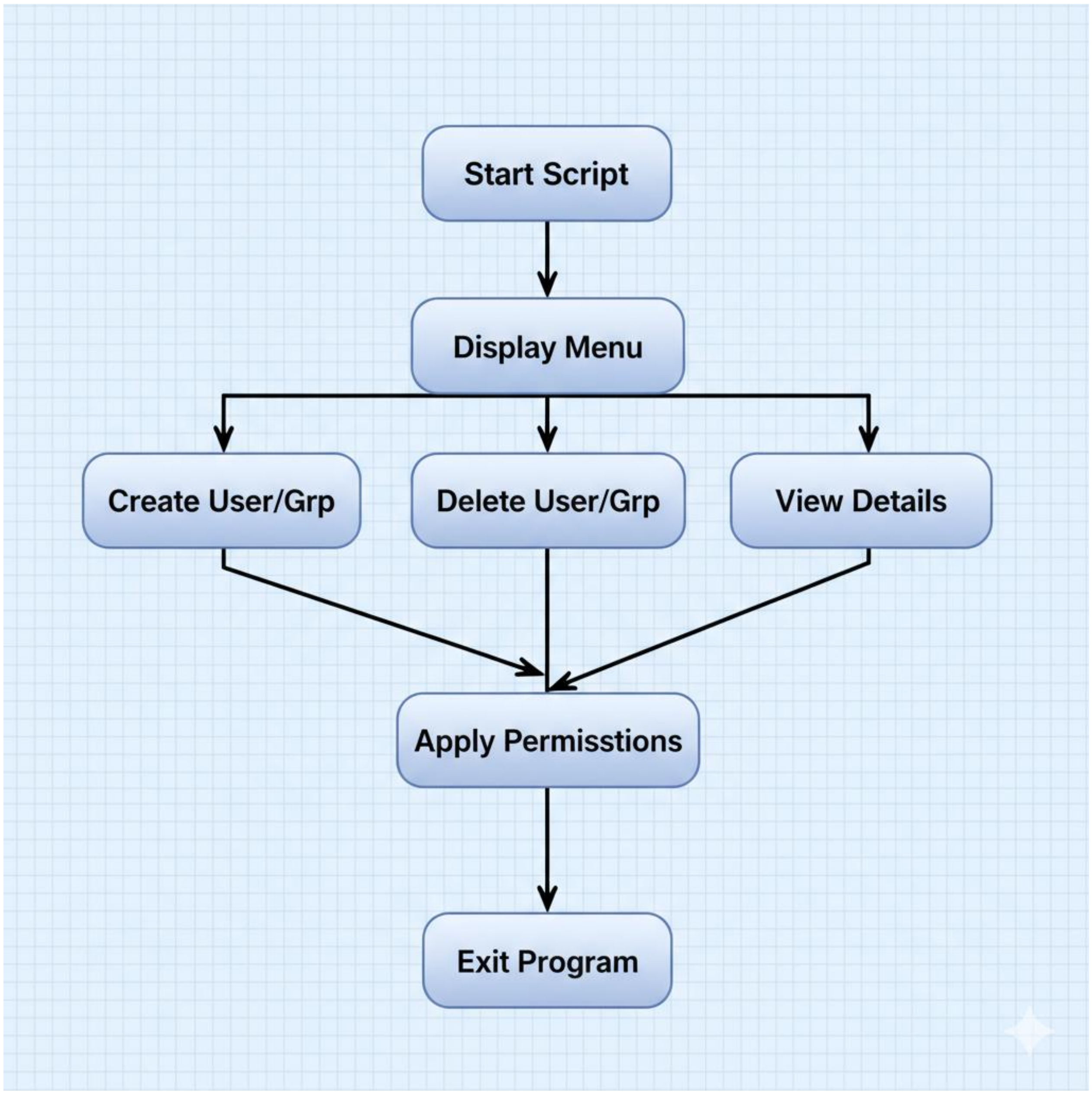
Tool / Command	Description
useradd	Creates a new user account
groupadd	Creates a new user group
usermod	Modifies existing user accounts
userdel	Deletes user accounts
groupdel	Deletes groups
chmod	Changes file or directory permissions
chown	Changes file or directory ownership
sudo	Executes administrative commands with elevated privileges
getent, id, cut	Used for fetching and displaying system information

6. METHODOLOGY:

The project is implemented using Bash scripting with a menu-driven interface. The script performs the following steps:

1. Displays a main menu with options for creating or deleting users and groups.
2. Uses conditional checks to prevent duplicate creation of users or groups.
3. Applies sudo for privileged operations like adding users, changing permissions, and creating directories.
4. Sets proper access control using the chmod and chown commands.
5. Continuously runs in a loop until the user selects the “Exit” option.

7. FLOWCHART


Start Script
Display Menu
Create User/Grp
Delete User/Grp
View Details
Apply Permisstions
Exit Program

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

8. IMPLEMENTATION

Script Name: user_group_menu_sudo.sh
Key Features:

- Menu-driven interface for easy navigation
- Uses sudo for safe administrative control
- Handles creation and deletion of users and groups
- Sets directory-level access control
- Displays system user/group information

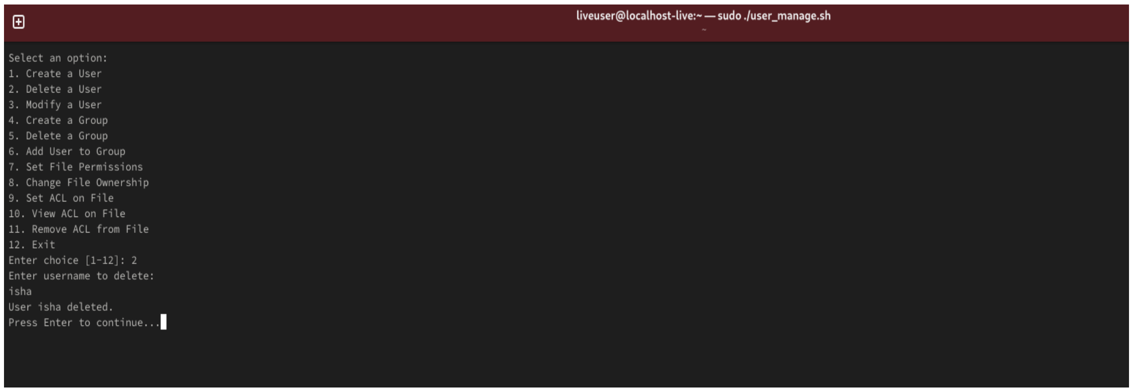
Code :

```
#!/bin/bash

# User Management Functions
create_user() {
   echo "Enter username to create: "
   read username
   sudo useradd $username
   echo "User $username created."
   sudo passwd $username
}

delete_user() {
   echo "Enter username to delete: "
   read username
   sudo userdel -r $username
   echo "User $username deleted."
}

modify_user() {
   echo "Enter username to modify: "
   read username
   echo "Enter new home directory (or leave blank to skip): "
   read homedir
   if [[ ! -z "$homedir" ]]; then
      sudo usermod -d $homedir $username
   fi
   echo "User $username modified."
}
```

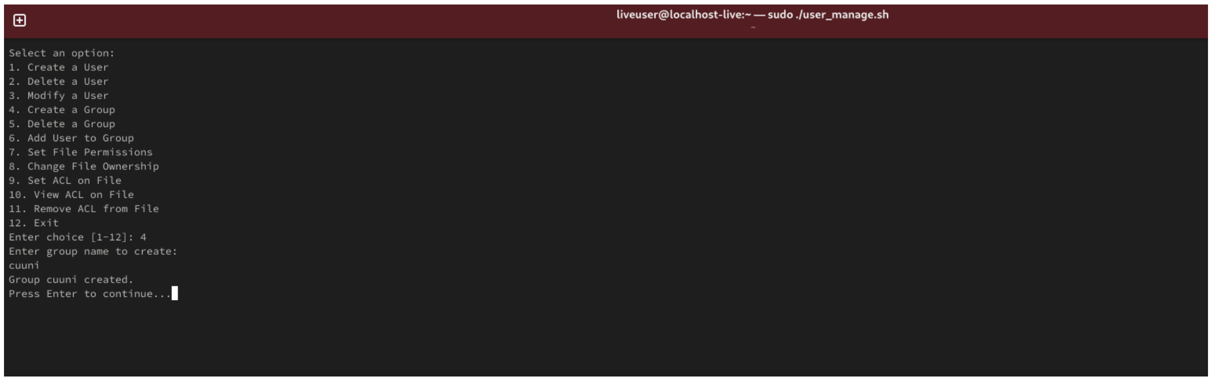
```
# Group Management Functions
create_group() {
   echo "Enter group name to create: "
   read groupname
   sudo groupadd $groupname
   echo "Group $groupname created."
}

delete_group() {
   echo "Enter group name to delete: "
   read groupname
   sudo groupdel $groupname
   echo "Group $groupname deleted."
}

add_user_to_group() {
   echo "Enter username to add to group: "
   read username
   echo "Enter group name: "
   read groupname
   sudo usermod -aG $groupname $username
   echo "User $username added to group $groupname."
}

# File Permissions Functions
set_file_permissions() {
   echo "Enter filename to set permissions: "
   read filename
   echo "Enter permissions (e.g., 755): "
   read permissions
   sudo chmod $permissions $filename
   echo "Permissions for $filename set to $permissions."
}

change_ownership() {
   echo "Enter filename to change ownership: "
   read filename
   echo "Enter new user ownership: "
   read user
   echo "Enter new group ownership: "
   read group
   sudo chown $user:$group $filename
echo "Ownership of $filename changed to $user:$group."
}

# ACL Management Functions
set_acl() {
    echo "Enter filename to set ACL: "
    read filename
    echo "Enter user to grant access: "
    read user
    echo "Enter permissions (e.g., rw): "
    read acl_perms
    sudo setfacl -m u:$user:$acl_perms $filename
    echo "ACL for $filename set to user $user with permissions $acl_perms."
}

view_acl() {
    echo "Enter filename to view ACL: "
    read filename
    getfacl $filename
}

remove_acl() {
    echo "Enter filename to remove ACL: "
    read filename
    echo "Enter user to remove ACL: "
    read user
    sudo setfacl -x u:$user $filename
    echo "ACL for user $user removed from $filename."
}

# Main Menu
while true; do
    clear
    echo "Select an option:"
    echo "1. Create a User"
    echo "2. Delete a User"
    echo "3. Modify a User"
    echo "4. Create a Group"
    echo "5. Delete a Group"
    echo "6. Add User to Group"
    echo "7. Set File Permissions"
    echo "8. Change File Ownership"
```

```
  echo "9. Set ACL on File"
    echo "10. View ACL on File"
    echo "11. Remove ACL from File"
    echo "12. Exit"
    read -p "Enter choice [1-12]: " choice

    case $choice in
        1) create_user ;;
        2) delete_user ;;
        3) modify_user ;;
        4) create_group ;;
        5) delete_group ;;
        6) add_user_to_group ;;
        7) set_file_permissions ;;
        8) change_ownership ;;
        9) set_acl ;;
        10) view_acl ;;
        11) remove_acl ;;
        12) echo "Exiting..."; break ;;
        *) echo "Invalid option, please try again." ;;
    esac
   read -p "Press Enter to continue..." dummy
done
```

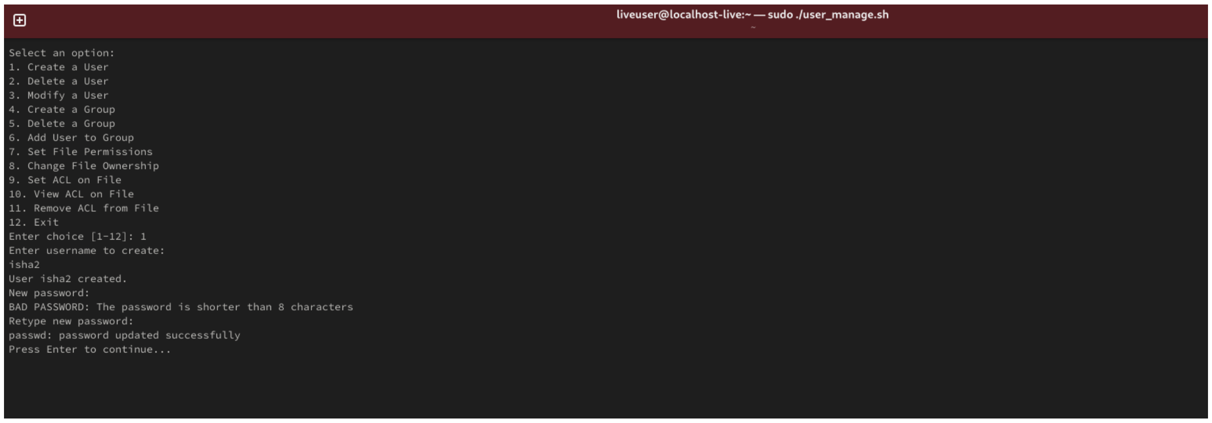
9. OUPUT:



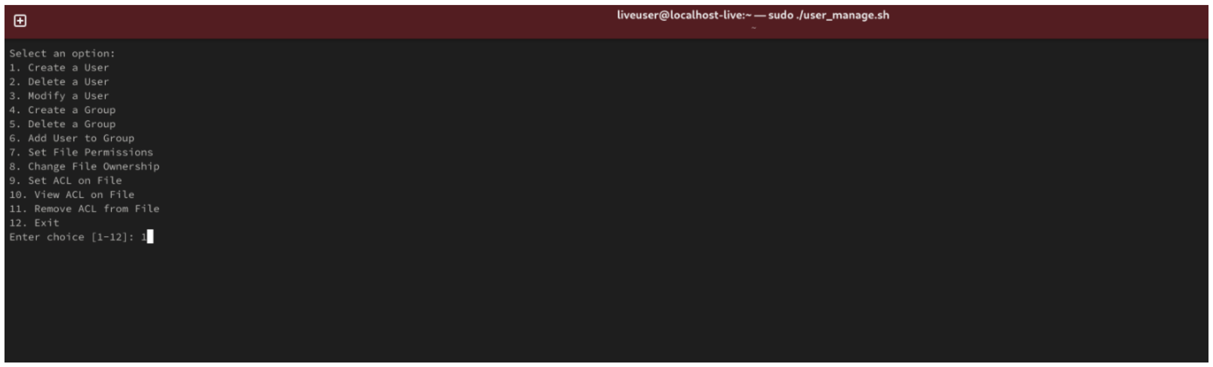
```
Select an option:
1. Create a User
2. Delete a User
3. Modify a User
4. Create a Group
5. Delete a Group
6. Add User to Group
7. Set File Permissions
8. Change File Ownership
9. Set ACL on File
10. View ACL on File
11. Remove ACL from File
12. Exit
Enter choice [1-12]: 2
Enter username to delete:
isha
User isha deleted.
Press Enter to continue...
```



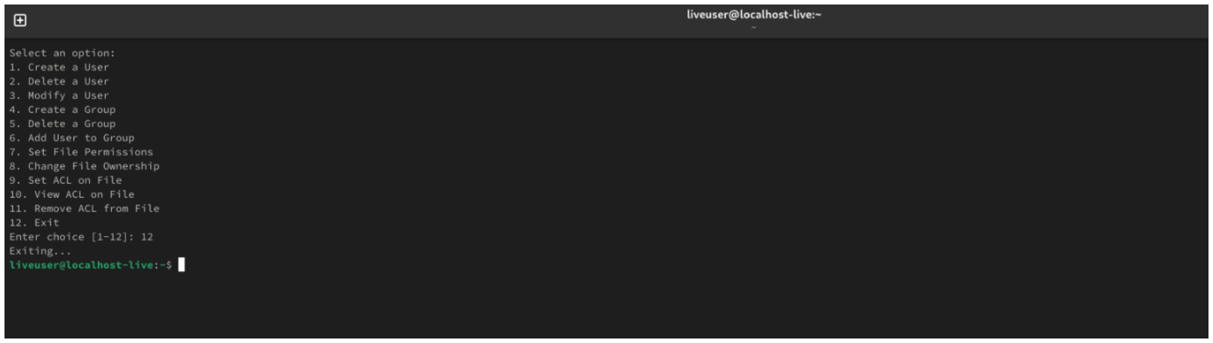
```
Select an option:
1. Create a User
2. Delete a User
3. Modify a User
4. Create a Group
5. Delete a Group
6. Add User to Group
7. Set File Permissions
8. Change File Ownership
9. Set ACL on File
10. View ACL on File
11. Remove ACL from File
12. Exit
Enter choice [1-12]: 4
Enter group name to create:
cuuni
Group cuuni created.
Press Enter to continue...
```



```
Select an option:
1. Create a User
2. Delete a User
3. Modify a User
4. Create a Group
5. Delete a Group
6. Add User to Group
7. Set File Permissions
8. Change File Ownership
9. Set ACL on File
10. View ACL on File
11. Remove ACL from File
12. Exit
Enter choice [1-12]: 1
Enter username to create:
isha2
User isha2 created.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
Press Enter to continue...
```



```
Select an option:
1. Create a User
2. Delete a User
3. Modify a User
4. Create a Group
5. Delete a Group
6. Add User to Group
7. Set File Permissions
8. Change File Ownership
9. Set ACL on File
10. View ACL on File
11. Remove ACL from File
12. Exit
Enter choice [1-12]: 1
```



```
Select an option:
1. Create a User
2. Delete a User
3. Modify a User
4. Create a Group
5. Delete a Group
6. Add User to Group
7. Set File Permissions
8. Change File Ownership
9. Set ACL on File
10. View ACL on File
11. Remove ACL from File
12. Exit
Enter choice [1-12]: 12
Exiting...
liveuser@localhost-live:~$
```

10. RESULT:

The script successfully:

- Automates user and group creation.
- Implements proper access control using Linux permissions.
- Provides a user-friendly, menu-based interface.
- Uses sudo securely without requiring full root login.

11. CONCLUSION

This project showcases the power and practicality of Bash scripting in automating routine administrative tasks within Linux systems. By leveraging the built-in capabilities of Bash and integrating them with essential system commands such as sudo, useradd, groupadd, and passwd, this script enhances both efficiency and system security.

The menu-driven interface simplifies complex tasks like user creation, group management, and permission allocation, making it accessible even for administrators with limited command-line experience. It allows for quick decision-making without requiring in-depth knowledge of Linux command syntax.

Incorporating sudo ensures that operations requiring elevated privileges are handled securely, preventing unauthorized access while still allowing flexibility for authorized users. This balance between usability and security is critical in multi-user environments, especially in professional or educational settings where system integrity must be preserved.

Overall, this project serves as an example of how automation via Bash scripting not only reduces administrative overhead but also minimizes the risk of human error. It paves the way for building more advanced and customizable tools that can be tailored to meet the specific needs of different Linux environments. With further enhancements like logging, error handling, or integration with GUI tools, this script can evolve into a full-fledged admin