Project Based Evaluation

Project Report

Semester-IV (Batch-2023)

Package Installer Menu

G19-PID-11

Supervised By:

Mrs. Parul Gulati

Submitted By:

Antriksh (2310991698)
Somya (23109923374)
Tamanna (2310992381)
Tanisha (2310992383)

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

ABSTRACT

This project introduces a menu-driven Bash script designed to simplify package management on Debian-based Linux systems. Traditional tools like apt and dpkg offer powerful functionality but can be intimidating for new users or inefficient for routine tasks. To address this, the script utilizes the dialog utility to present a graphical, text-based interface in the terminal, enabling easier navigation and operation.

The script allows users to install or remove both predefined and custom packages, check the installation status of software, view detailed package information, and maintain a log of all actions. Additional features include monitoring system activity by displaying currently logged-in users and active processes, as well as the ability to back up log files for future reference. These functionalities make the tool useful not only for casual users but also for system administrators looking to streamline repetitive tasks.

By bridging the gap between command-line complexity and user-friendly design, this project improves accessibility, encourages learning, and enhances productivity in Linux environments.

Table of Contents

1. Introduction

1.1 Background and Motivation

Linux-based systems are known for their flexibility and powerful package management capabilities. However, for new users or administrators, using the terminal to install, remove, or check software packages can be intimidating. This project aims to bridge that usability gap by providing a menu-driven interface for managing packages using dialog and standard Linux package management commands (apt, dpkg).

The motivation behind this tool is to simplify routine administrative tasks by wrapping complex command-line operations within an intuitive and user-friendly interface, ultimately improving efficiency and reducing the risk of command errors.

1.2 Objectives of the Project

The primary objective of this project is to design and implement a dialog-based, menu-driven Bash script that simplifies package management operations on Debian-based Linux systems. This tool is intended to provide a user-friendly interface for performing administrative tasks that are typically done via command-line, making them more accessible and less error-prone.

The key objectives include:

1.Simplified Package Management

Allow users to install, remove, and check the status of software packages without manually entering complex commands.

2.Interactive Graphical Interface

Use the dialog utility to build an intuitive text-based GUI for better user interaction.

3.Automated Logging

Maintain a detailed log of all operations (installations, removals, errors) to support auditing and troubleshooting.

4.Information Access

Provide real-time access to package information, currently logged-in users, and active processes.

5.Backup and Recovery Support

Enable users to back up logs and support minimal recovery actions in case of accidental errors.

6.Educational Purpose

Help students and beginners in Linux administration learn how shell scripting and system tools can be used in automation.

1.3 Scope of the Work

The scope of this project is limited to the development and execution of a Bash-based, dialog-driven script that assists in basic package management tasks on Debian-based Linux systems. The script utilizes core Linux utilities such as apt, dpkg, and dialog to create a menu-driven interface for system administrators.

Specifically, the script provides the following functionalities:

1.Package Installation: Prompts the user to enter a package name and installs it using apt-get Install.

2.Package Removal: Allows the user to remove installed packages using apt-get purge, after verifying that the package exists.

3.Status Check: Checks whether a given package is currently installed using dpkg -l.

4.Package Information: Displays detailed information about a specified package using apt-cache show.

5.Log Viewing and Backup: Maintains an installation/removal log in /var/log/package_installer.log and allows backup to /tmp/package installer/backup.log.

6.System Monitoring: Displays currently logged-in users (who) and top memory-consuming processes (ps).

7.Dialog-Based GUI: All user interactions are handled via the dialog utility, creating a text-based graphical interface in the terminal.

8.Root Privilege Enforcement: Ensures the script runs with root privileges to avoid permission issues.

The script is intended to operate in a CLI (Command-Line Interface) environment without the need for a full desktop GUI. It is suitable for lightweight administrative tasks, educational demonstrations, or beginner-friendly system automation.

1.4 Report Structure

This report is organized into well-defined sections to present a clear and comprehensive view of the development and execution of the dialog-based package installer system. The structure of the report is as follows:

Chapter 1 – Introduction:

Provides the background, objectives, motivation, and scope of the project, along with a summary of the report's organization.

Chapter 2 – System Environment:

Details the hardware and software setup, Linux distribution used, and supporting tools and utilities like dialog, apt, and dpkg.

Chapter 3 – Conceptual Overview:

Describes the key Linux concepts, system components, commands, and services relevant to package management and system monitoring.

Chapter 4 – UML Diagrams:

Contains use case, activity, and sequence diagrams to represent the system's functionality and user interaction flow.

Chapter 5 – Implementation Details:

Provides an in-depth explanation of the script’s functionality, including the menu system, code flow, major functions, and their outputs. Also includes screenshots of the interface.

Chapter 6 – Security and Optimization:

Discusses the root privilege check, log maintenance, and performance considerations implemented in the script.

Chapter 7 – Testing and Validation:

Covers the test cases executed, expected results, troubleshooting steps, and validation using logs.

Chapter 8 – Challenges and Limitations:

Highlights the technical difficulties faced during scripting, how they were resolved, and the limitations of the current system.

Chapter 9 – Conclusion and Future Work:

Summarizes the outcomes of the project, lessons learned, and possible future enhancements such as GUI integration or advanced logging.

Chapter 10 – References:

Lists the documentation, commands, and tutorials consulted during the development process.

Chapter 11 – Appendices:

Contains the complete source code, screenshots, and any additional configuration or supporting files.

2. System Environment

2.1 Hardware and Software Requirements

The project was developed and tested on a system with the following minimum hardware and software specifications:

Processor: Dual-core CPU (1.5 GHz or higher)

Memory: 2 GB RAM (minimum), 4 GB or more recommended

Storage: At least 5 GB of free disk space

Operating System: Debian-based Linux distribution (e.g., Ubuntu)

Privileges: Root or Sudo access is required to execute package management operations

These specifications ensure sufficient performance to run Bash scripts, manage packages, and display dialog interfaces smoothly.

2.2 Linux Distribution and Version

The project was implemented on **Ubuntu 22.04.2 LTS**, a widely-used Debian-based distribution known for its stability and support. However, the script is compatible with other Debian-based systems such as Debian, Linux Mint, and Kali Linux, provided they support the `apt` and `dialog` utilities.

2.3 Tools and Utilities Used

The following tools and utilities were essential to the development and operation of this project:

Bash: The primary scripting language used for automation and logic implementation.

dialog: A utility that provides a terminal-based graphical interface for user interaction.

apt: Advanced Package Tool used to handle package installation and removal.

dpkg: Debian Package tool used to query and manage installed packages.

ps, who and other native Linux commands: Used for system monitoring features such as viewing running processes \and active users.

echo, awk, grep, head, and date: Utilities used for text processing and formatting within the script.

These components collectively form the backbone of the interactive package management system designed in this project.

3. Conceptual Overview

3.1 Key Concepts Related to the Project

This project centres around the concept of simplifying Linux package management through a user-friendly interface. Traditionally, tools like apt and dpkg are used from the command line, which may be intimidating for users unfamiliar with terminal operations. By introducing a menu-driven interface using the dialog utility, the project abstracts complex commands into intuitive options. Additionally, the project emphasizes automation, usability, and system monitoring—key principles that align with modern system administration practices.

3.2 Relevant System Components and Files

The script interacts with several system-level components and directories, including:

/var/log/package_installer.log: Custom log file used to record installation and removal operations.

/tmp/package_installer/: Temporary directory created at runtime for storing intermediate data and backups.

/etc/apt/: Directory containing APT configuration files and repositories (used indirectly by apt commands).

/usr/bin/apt, /usr/bin/dpkg: Executables for the core package management tools.

Environment variables and standard I/O streams are also used for capturing user input and displaying results.

These components enable the script to function reliably across sessions and provide persistent, organized data handling.

3.3 Linux Commands and Services Involved

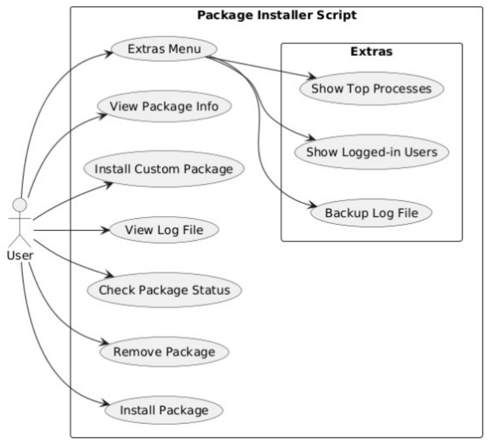
Several core Linux commands and services are utilized in this project

Command/Service	Description
apt-get	Manages packages on Debian-based systems. Used for installing, upgrading, and removing software packages.
dpkg -l	Lists all installed packages, useful for checking whether a specific package is installed.
apt-cache show	Displays detailed information about a package, including its version, dependencies, and description.
dialog	Creates a terminal-based GUI for interactive menus, input boxes, message boxes, and text areas.
ps	Displays information about active processes on the system. Often used to show running processes.
who	Shows information about logged-in users, including their usernames and terminal details.
cp	Copies files or directories from one location to another.
rm	Removes files or directories.
touch	Creates a new empty file or updates the timestamp of an existing file.
mkdir	Creates new directories.
awk	A powerful text processing tool used for filtering and formatting data.
grep	Searches for specific patterns in text and outputs matching lines.
head	Outputs the first few lines of a file or command result.

4. UML Diagrams

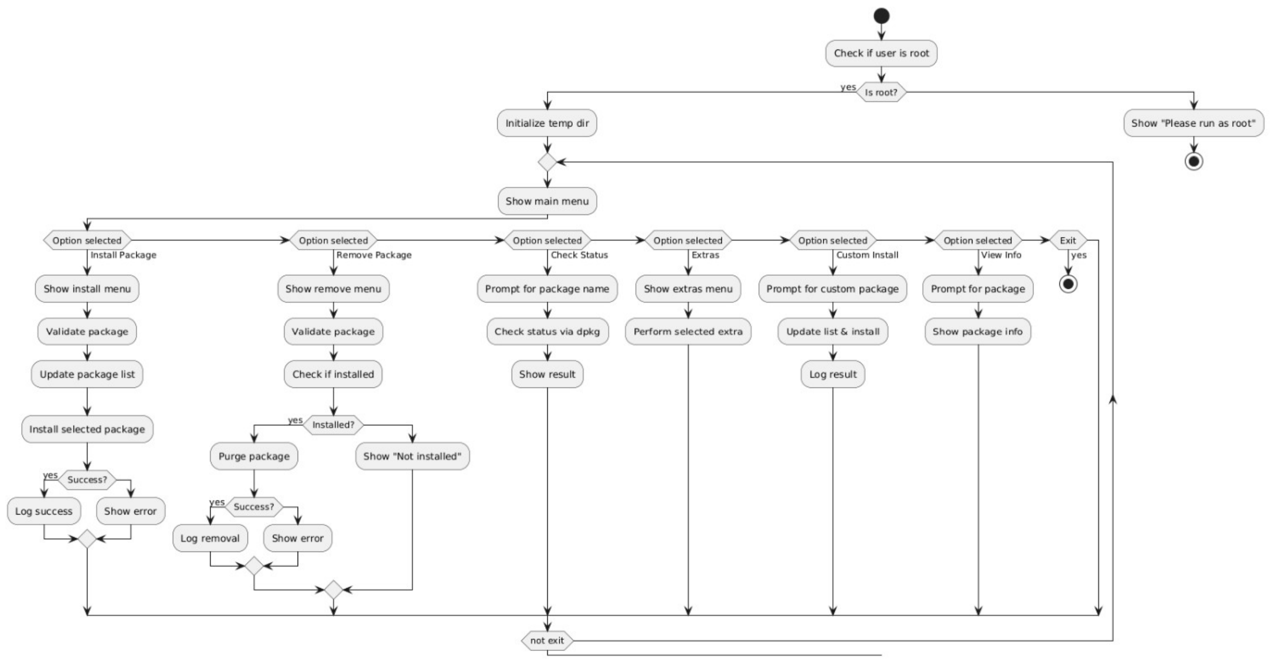
4.1 Use Case Diagram

Shows the interactions between the system administrator and the script. Main use cases include installing/removing packages, checking status, viewing logs, and performing extra system tasks.



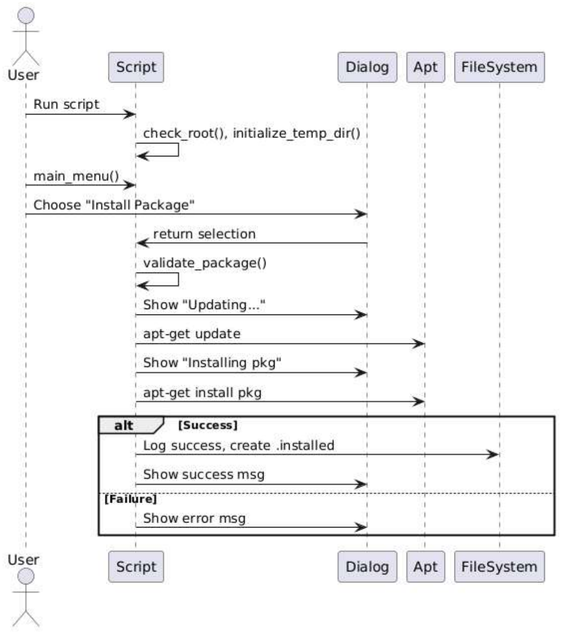
4.2 Activity Diagram

Depicts the step-by-step workflow of key operations like installing a package. Includes user input, validation, execution of commands, logging, and user feedback.



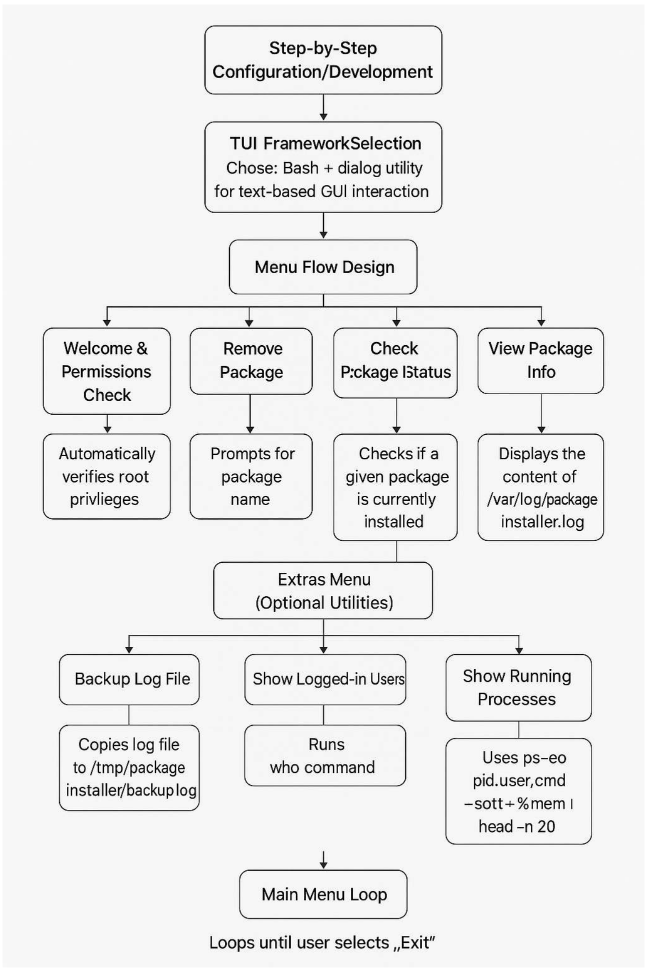
4.3 Sequence Diagram

Illustrates the interaction flow between components like the menu, validator, APT executor, and logger during operations such as package installation or removal.



5. Implementation Details

5.1 Step-by-Step Configuration/Development


Step-by-Step Configuration/Development
TUI FrameworkSelection
Chose: Bash + dialog utility for text-based GUI interaction
Menu Flow Design
Welcome & Permissions Check
Remove Package
Check P:ckage IStatus
View Package Info
Automatically verifies root privlieges
Prompts for package name
Checks if a given package is currently installed
Displays the content of /var/log/package installer.log
Extras Menu (Optional Utilities)
Backup Log File
Show Logged-in Users
Show Running Processes
Copies log file to /tmp/package installer/backup log
Runs who command
Uses ps-eo pid.user,cmd –sott+%mem I head –n 20
Main Menu Loop
Loops until user selects „Exit"

TUI Framework Selection

For the terminal-based package management utility, the selected TUI framework is Bash with the dialog utility.

Reasons for choosing dialog:

- Pre-installed on many Linux distributions.

- Lightweight and well-suited for terminal-based automation.
- Provides interactive components such as menus, input boxes, and message dialogs.

An alternative tool, whiptail, was also considered due to its simplicity and similar syntax, but dialog offered broader capabilities and compatibility for this implementation.

Menu Flow Design

1. Welcome & Permissions Check

- The script begins by checking whether it is being run as the root user.
- If not run as root, a dialog error message is displayed and the script exits immediately.
- This step ensures that commands like apt, dpkg, and system directory access are permitted.

2. Temporary Directory Setup

A working directory /tmp/package_installer is created (if not already existing).

This directory is used to store:

- Temporary command outputs.
- Log files.
- . installed markers to track installed packages.

3. Install Package

- The user is prompted to input the desired package name using a dialog input box.
- The script runs apt-get update to refresh package lists.
- Then it attempts to install the package using apt-get install -y <package>, capturing all output.

On success:

- The event is logged with a timestamp.

- installed marker file is created for tracking.

On failure:

- A dialog error message is shown to the user.

4. Remove Package

The user is prompted for a package name to uninstall.

If installed, it runs apt-get purge -y <package> and logs the result.

On success:

- The log is updated.
- The associated. installed marker is deleted (if it exists).

On failure:

- An error dialog is shown.

5. Check Package Status

- Verifies if a package is currently installed.
- Uses a combination of dpkg -l and grep to check installation status.
- Results are displayed in a dialog box.

6. View Package Info

Retrieves metadata and details about a given package using: **apt-cache show <package>**

- The information is presented in a scrollable textbox.
- If the package does not exist, the script handles it gracefully and informs the user.

7. View Log File

Opens and displays the contents of **/var/log/package_installer.log.**

The log is presented in a scrollable dialog textbox.

8. Main Menu Loop

Presents users with a looping menu that offers the following options:

- Install
- Remove
- Check Status
- View Log
- View Info
- Extras
- Exit

The loop continues until the user selects "Exit".

Error Handling Implementation

- Empty Input Validation: Checks if the package name is provided before proceeding with install/remove operations.
- Missing Tool Detection: Ensures the presence of essential tools (dialog, apt, dpkg). If any are missing, the script exits with a meaningful error.
- Command Failures: All command outputs (including errors) are logged. Users are shown descriptive error dialogs when operations fail.
- File Handling Safety: Before reading from or writing to log or marker files, the script checks for file existence and appropriate permissions.

5.2 Commands and Scripts Used

The package installer system was implemented as a Bash script utilizing the dialog utility for a terminal-based GUI. The script supports package installation, removal, status checking, log viewing, package information retrieval, and several extras for system insight.

Key Commands and Utilities:

Command	**Purpose**
dialog	Used for input boxes, message boxes, menus, and displaying text in dialogs
apt-get	Installs (install) and removes (purge) packages using APT
dpkg -l	Lists installed packages; used to verify package installation status
apt-cache show	Retrieves metadata and detailed information about a given package
who	Displays currently logged-in users
ps -eo pid, user, cmd --sort=-%mem	Lists processes sorted by memory usage (top memory consumers)

Script Flow Summary:

1.Initialization:

- Verifies if the user is root using $EUID.
- Creates a temporary working directory: /tmp/package_installer.

2. Installation Logic:

- Prompts the user for a package name.
- Updates the package list with apt-get update.
- Installs the package using apt-get install -y.
- Logs the outcome and creates a. installed marker.

3. Removal Logic:

- Prompts for a package name.
- Verifies if the package is installed using dpkg -l.
- Uses apt-get purge -y to uninstall and logs the action.

4. Status Check:

- Uses dpkg -l with grep to determine if the package is installed.

5. Information Retrieval:

- Uses apt-cache show to fetch package details.
- Displays information using a scrollable textbox.

6. Log Management:

- Displays contents of the log file: /var/log/package_installer.log.
- Backs up logs to /tmp/package_installer/backup.log.

7. System Insights:

- Displays logged-in users using who.
- Shows the top 20 processes by memory usage with ps.

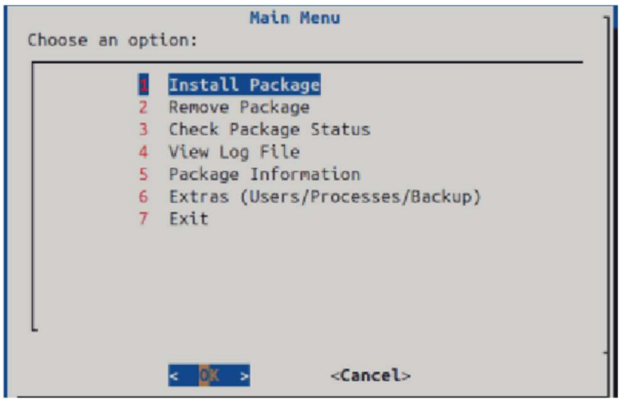
8. Menu Navigation:

- Implements an interactive loop with options:
- Install, Remove, Status, Log, Info, Extras, Exit.

5.3 Screenshots and Outputs

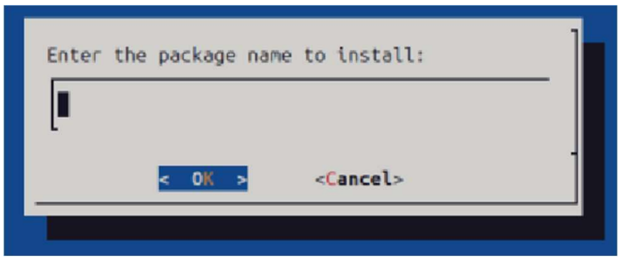
Below are representative screenshots and descriptions of key interactions with the system. These demonstrate usability and feature completeness.

1.Main Menu Interface



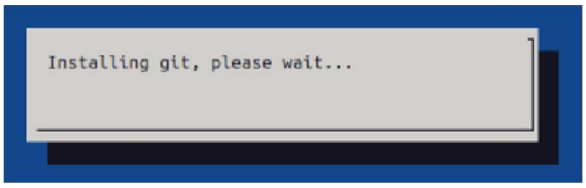
```
Main Menu
Choose an option:

  1  Install Package
  2  Remove Package
  3  Check Package Status
  4  View Log File
  5  Package Information
  6  Extras (Users/Processes/Backup)
  7  Exit

      <  OK  >          <Cancel>
```

2. Install Package Menu



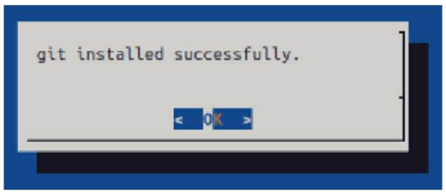
```
Enter the package name to install:


      <  OK  >          <Cancel>
```

3. Installation in Progress Dialog



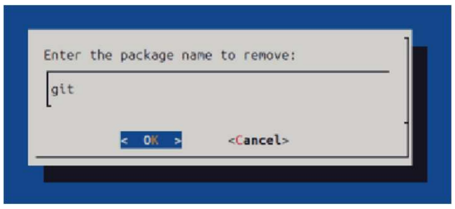
```
Installing git, please wait...
```

4. Successful Installation Message

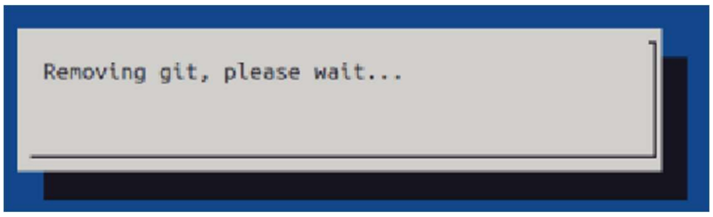


```
git installed successfully.

< OK >
```

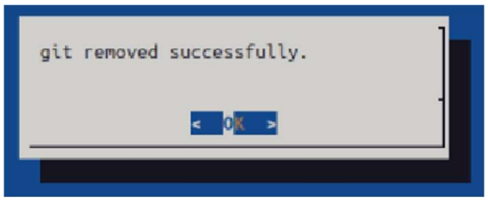
5. Remove Package Menu



```
Enter the package name to remove:
git

< OK >        <Cancel>
```

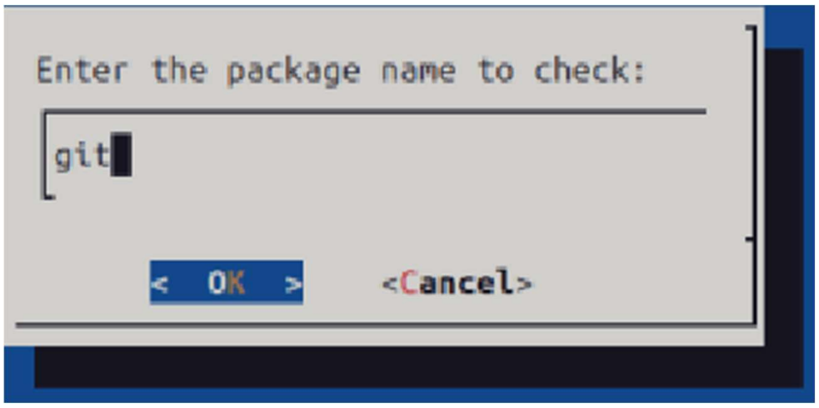


```
Removing git, please wait...
```

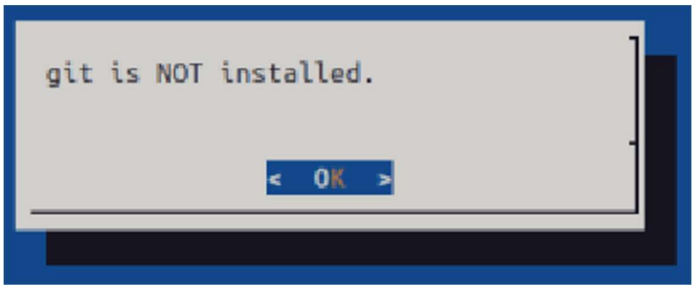


```
git removed successfully.

< OK >
```

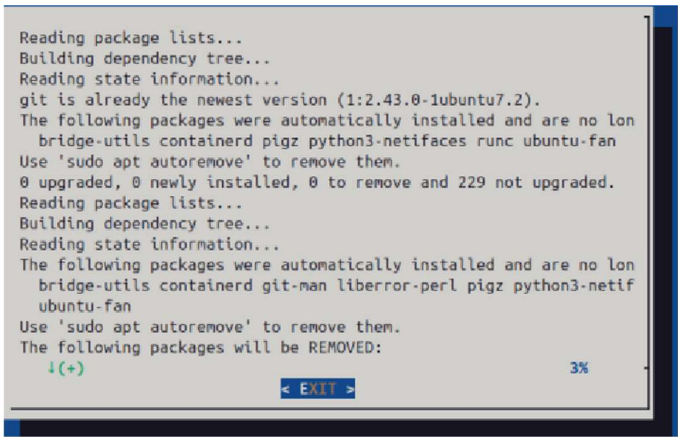
6. Check Package Status



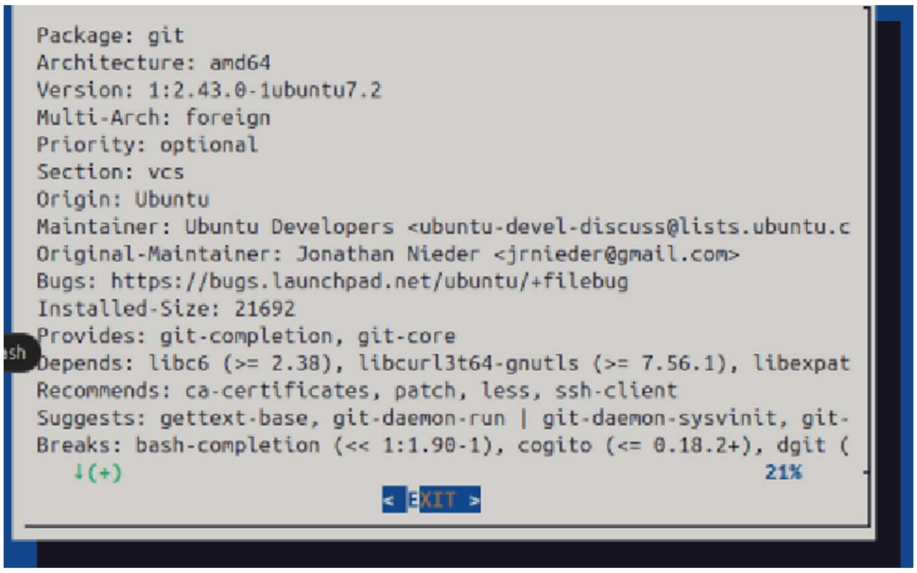
```
Enter the package name to check:
git

< OK >        <Cancel>
```


```
git is NOT installed.
```

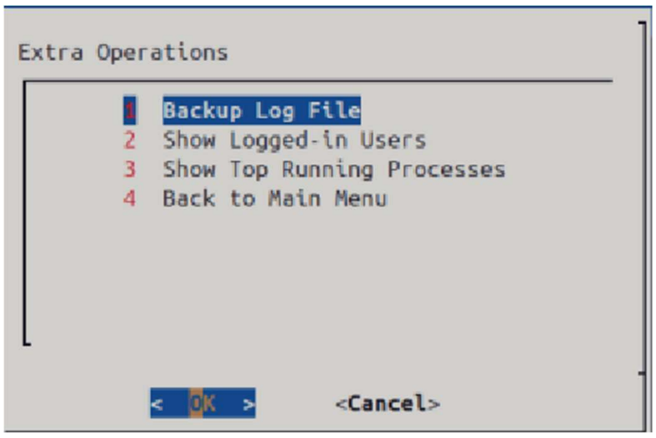
7. View Installation Log File


```
Reading package lists...
Building dependency tree...
Reading state information...
git is already the newest version (1:2.43.0-1ubuntu7.2).
The following packages were automatically installed and are no lon
  bridge-utils containerd pigz python3-netifaces runc ubuntu-fan
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 229 not upgraded.
Reading package lists...
Building dependency tree...
Reading state information...
The following packages were automatically installed and are no lon
  bridge-utils containerd git-man liberror-perl pigz python3-netif
  ubuntu-fan
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
   ↓(+)                                                    3%
```

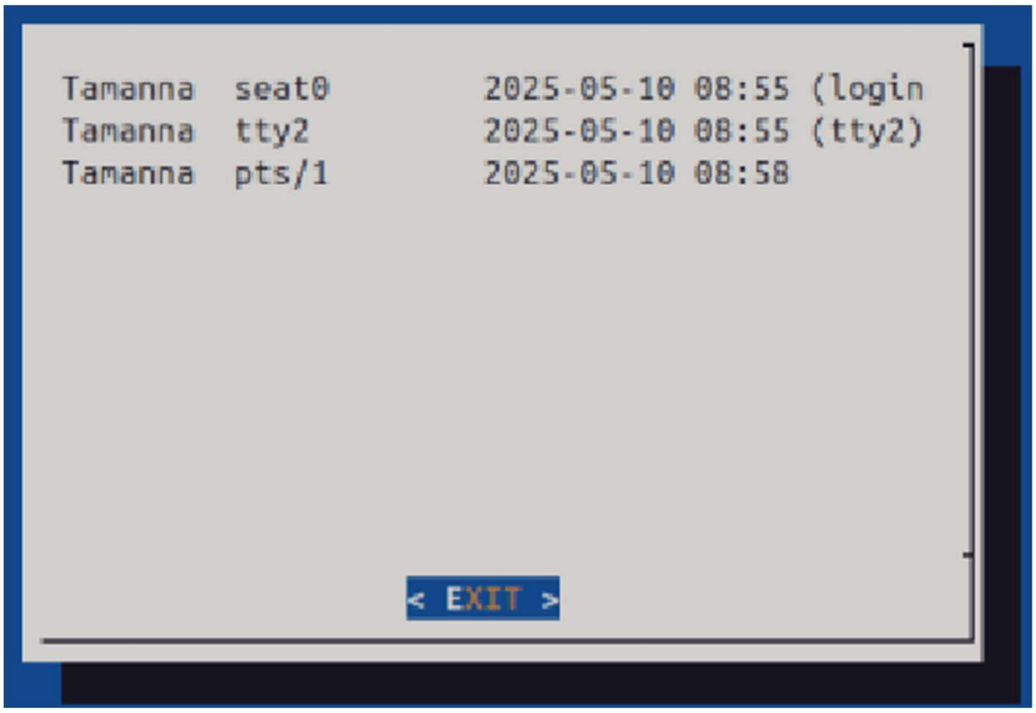
8. View Package Info


```
Package: git
Architecture: amd64
Version: 1:2.43.0-1ubuntu7.2
Multi-Arch: foreign
Priority: optional
Section: vcs
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.c
Original-Maintainer: Jonathan Nieder <jrnieder@gmail.com>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 21692
Provides: git-completion, git-core
Depends: libc6 (>= 2.38), libcurl3t64-gnutls (>= 7.56.1), libexpat
Recommends: ca-certificates, patch, less, ssh-client
Suggests: gettext-base, git-daemon-run | git-daemon-sysvinit, git-
Breaks: bash-completion (<< 1:1.90-1), cogito (<= 0.18.2+), dgit (
   ↓(+)                                                    21%
```

9. Extras Menu (System Tools)

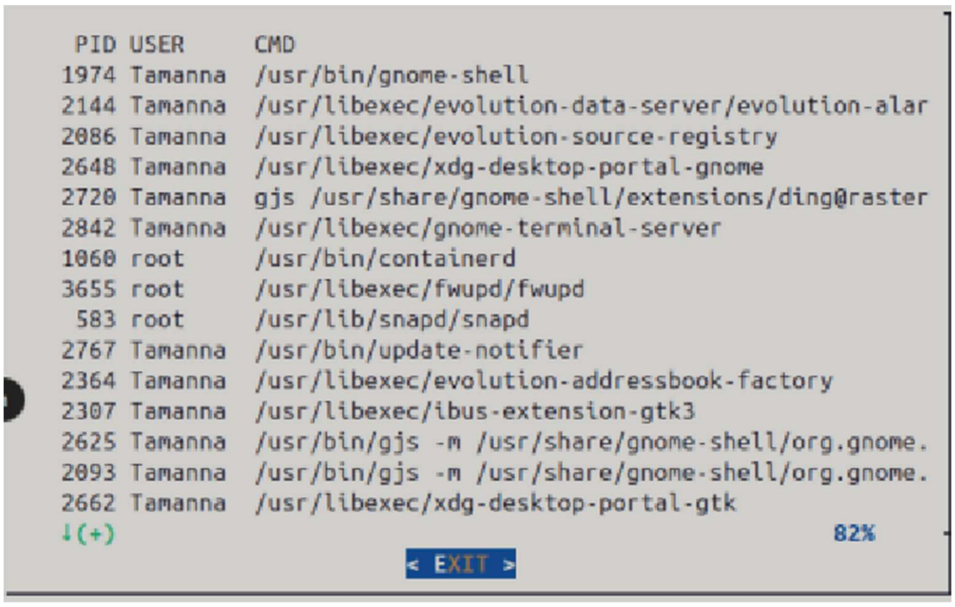


10. Show Logged-in Users



```
Tamanna  seat0        2025-05-10 08:55 (login
Tamanna  tty2         2025-05-10 08:55 (tty2)
Tamanna  pts/1        2025-05-10 08:58
```

11. Show Top Running Processes



```
  PID USER      CMD
 1974 Tamanna  /usr/bin/gnome-shell
 2144 Tamanna  /usr/libexec/evolution-data-server/evolution-alar
 2086 Tamanna  /usr/libexec/evolution-source-registry
 2648 Tamanna  /usr/libexec/xdg-desktop-portal-gnome
 2720 Tamanna  gjs /usr/share/gnome-shell/extensions/ding@raster
 2842 Tamanna  /usr/libexec/gnome-terminal-server
 1060 root     /usr/bin/containerd
 3655 root     /usr/libexec/fwupd/fwupd
  583 root     /usr/lib/snapd/snapd
 2767 Tamanna  /usr/bin/update-notifier
 2364 Tamanna  /usr/libexec/evolution-addressbook-factory
 2307 Tamanna  /usr/libexec/ibus-extension-gtk3
 2625 Tamanna  /usr/bin/gjs -m /usr/share/gnome-shell/org.gnome.
 2093 Tamanna  /usr/bin/gjs -m /usr/share/gnome-shell/org.gnome.
 2662 Tamanna  /usr/libexec/xdg-desktop-portal-gtk
 ↓(+)                                                   82%
                         < EXIT >
```

6. Security and Optimization

6.1 Hardening Measures Taken

The Package Installer System incorporates various security practices to ensure safe execution of privileged operations and to protect system integrity:

Principle of Least Privilege

• The script is designed to be executed only by root or with Sudo, ensuring that only users with administrative privileges can perform sensitive package operations.

• No excessive permission escalation is embedded into the script; only the required level of access is invoked when needed.

Input Validation

• All user inputs (like package names) are validated to ensure they are not empty, preventing malformed command execution.

• Special characters and blank submissions are handled using conditional checks to avoid command injection or unintentional behaviour.

Secure Logging

•The script logs all installation and removal events with timestamps to /var/log/package_installer.log, which is a system-level log file typically protected with elevated access permissions.

•All log operations append data, avoiding overwrites or data loss.

Temporary File Handling

• All temporary files and user session data are stored in /tmp/package_installer, which is automatically cleaned up on most Linux systems during reboot.

• Sensitive information such as session history or user operations are not exposed to non-root users.

Restricted Operations

• The use of dialog, apt, and dpkg ensures that only predefined, tested operations are available to users. Arbitrary command execution is not possible through the interface.

6.2 Performance Tuning and Efficiency

The script is optimized for fast and efficient package management on Debian-based systems while maintaining a smooth user experience:

Resource Management

•Background operations like apt-get update and package installations are run silently in the background and logged, avoiding resource-intensive UI blocking.

•Lightweight tools (ps, who, dpkg, etc.) are used for system information retrieval.

Execution Optimization

• The system introduces brief intentional delays (sleep 1) during key actions to simulate processing and avoid overwhelming the user with rapid screen changes.

• Checks are made before performing installations or removals to avoid redundant actions (e.g., confirming whether a package is installed before attempting removal).

Dialog UI Responsiveness

• Dialog screens provide asynchronous user interaction and quick feedback during operations (e.g., info boxes showing progress during installations).

• The UI is kept minimal and intuitive to maintain responsiveness on low-resource systems.

6.3 Backup and Recovery Measures

To ensure resilience and data preservation, the script includes basic but effective backup and recovery mechanisms:

Log File Backup

• The user can create a backup of the installation log file, stored as backup.log in the temporary directory.

• This ensures that a copy of important system activity is preserved in case the original log becomes inaccessible or corrupted.

State Tracking

• Installed package state is tracked through temporary. installed files, which can assist in creating restoration logic in future versions.

• Although not currently auto-restorative, this foundation supports the extension of rollback or reinstallation features.

Temporary File Cleanup

• Files in /tmp/package_installer are recreated upon each script execution, ensuring stale data does not persist across sessions.

• The modular structure allows future enhancements like session state recovery or persistent configuration snapshots.

7. Testing and Validation

7.1 Test Scenarios and Expected Results

The script has been tested under various operational conditions to ensure reliable performance

Test Scenario	**Method**	**Expected Result**
Root Permission Check	Run without Sudo or as non-root user	Dialog displays warning and exits gracefully
Package Installation	Enter a valid package name (e.g., curl)	Package installs, log entry created, success message
Empty Package Input	Submit blank package name	User is prompted with validation message
Remove Uninstalled Package	Attempt to remove a package not installed	Dialog warns user that package is not installed
View Log File	Use log viewing option after installation	Shows recent actions with timestamps
Check Installed Package Status	Query for an existing system package	Confirms presence using dpkg -l
Package Information View	Input package name (e.g., git)	Shows metadata using apt-cache show
View System Users	Access "Extras" and select user display	Lists currently logged-in users
Top Processes	Open "Extras" and view processes	Displays top memory-consuming processes

7.2 Troubleshooting Techniques

The script supports troubleshooting through built-in utilities and validation checks:

Diagnostic Steps

1.Installation Failure – If a package fails to install, the user is notified, and error logs are stored for reference.

2.Empty Fields – Blank inputs are blocked early, preventing command failure.

3.Non-existent Packages – Checked using apt-cache show, returning a "not found" message.

4.Permission Errors – Enforced root-level access with appropriate dialog alerts.

Isolation Techniques

•The use of modular dialogs (menu-based flow) helps isolate each function, making it easier to identify failing steps.

•Logs enable after-action review and help track the exact command that failed.

Error Prevention Strategies

• Input validation and conditional branching prevent execution of invalid commands.

• Use of system utilities like dpkg -l prevents unnecessary or dangerous operations.

7.3 Logs and Monitoring Tools

Logging and session tracking are essential features of the Package Installer System:

Log File Details

- File Path: /var/log/package_installer.log
- Contents: Date-stamped entries for all install/remove operations
- Format: YYYY-MM-DD HH:MM: SS: Action Package Name (e.g., 2025-05-08 14:30:12: Installed git)

Monitoring Commands Used

Utility	Purpose	Usage
who	List logged-in users	Captured in extras menu
ps	Show top processes by memory usage	Sorted and displayed in a dialog
dpkg -l	Check installed packages	Used for status checks/removal
apt-cache show	View package info	Used in Package Information

Session Continuity and Visibility

• Logs enable tracking across sessions even after a reboot.

• Backup options allow duplication of logs for report or audit purposes.

Suggested Enhancements (for future)

• Implement auto-log rotation to prevent excessive growth.

• Integrate real-time log monitoring or alerts via system notifications.

8. Challenges and Limitations

8.1 Problems Faced During Implementation

During the development of the dialog-based Package Installer System, several challenges were encountered:

1. Ensuring Root Privileges:

Many commands like apt-get install, purge, and log file writing to /var/log/ require superuser access. Initially, the script failed when executed without root, so a root-check function had to be implemented at the start.

2. Handling User Input from Dialog Boxes:

Capturing user input using dialog with correct redirection (3>&1 1>&2 2>&3) was tricky, especially when reading inputs and returning values from multiple menus. It required trial and error to manage standard input/output properly.

3. Log File Permission Issues:

Writing logs to /var/log/package_installer.log caused permission errors if not run as root. This necessitated verifying log file access and managing backup creation in a temporary directory (/tmp).

4. Package Validation Before Removal:

Attempting to remove non-installed packages led to unnecessary errors. To prevent this, the script had to validate package status using dpkg -l and grep, which required parsing command outputs carefully.

5. Dialog Layout Adjustments:

Setting appropriate dialog box dimensions (e.g., height, width) was important to prevent content cutoff. Some screens required adjustment after testing with different terminal sizes.

6. Limited Feedback During Long Operations:

Commands like apt-get update or install take time but don’t show output directly in dialog. The use of --infobox was a workaround, but it doesn't display real-time feedback, which may confuse users during long operations.

7. Inconsistent Behavior Across Systems:

Testing the script on different Debian-based systems revealed minor differences in apt behavior and default permissions, requiring the script to be tested and adjusted accordingly.

8. Displaying Top Processes Accurately:

The ps command output varies in length and format. Trimming it using head -n 20 and displaying the most useful columns (pid, user, cmd) required experimenting with various formatting options.

8.2 Workarounds and Fixes

To address the problems encountered during the implementation of the dialog-based Package Installer System, several workarounds and fixes were applied:

1. Root Privilege Enforcement:

A check_root() function was implemented at the beginning of the script to terminate execution with a dialog alert if the script is not run as the root user. This ensures administrative commands execute without permission issues.

2. Reliable Dialog Input Handling:

The use of 3>&1 1>&2 2>&3 redirection was standardized across all dialog --inputbox and --menu calls to ensure accurate capture of user inputs and menu selections.

3. Log File Backup to /tmp:

Instead of risking permission issues with backup operations in /var/log/, log backups are stored in the temporary working directory (/tmp/package_installer). This avoids write-permission errors and keeps the log accessible for all users during runtime.

4. Package Status Validation:

To prevent errors during removal of uninstalled packages, a validation step using dpkg -l and grep was added. Only if the package is listed as installed will the apt-get purge command proceed.

5. Fixed Dialog Dimensions:

Dialog box dimensions (e.g., 20x70 for textboxes, 6x40 for message boxes) were standardized based on testing in typical terminal windows. This minimized layout issues and ensured readability.

6. Infoboxes for Process Feedback:

Although real-time progress isn't visible, dialog --infobox was used to at least inform users of ongoing operations like installation, updates, or removal. A short sleep was added to ensure the **infobox is visible before longer processes start.**

7. Conditional File Checks:

Functions like view_log_file, backup_log, and package_info now check if target files exist before attempting to read or copy them, reducing the chance of errors or dialog crashes.

8. Trimmed Output for Processes and Users:

The output of commands like ps and who was redirected to temporary files and truncated using head to fit into dialog boxes properly without overflow, ensuring clean and readable output.

8.3 Known Issues or Constraints

Despite successfully implementing the core functionalities, the Package Installer System has a few known limitations and constraints:

1. Limited to Debian-Based Systems:

The script relies on apt-get, dpkg, and apt-cache, which are specific to Debian-based Linux distributions (e.g., Ubuntu, Kali). It will not work on Red Hat, Arch, or other non-Debian systems without major modifications.

2. No Dependency Resolution Display:

While packages are installed via apt-get, the script does not display or warn about additional dependencies that may be installed or removed as part of the process.

3. Lack of Real-Time Installation Progress:

The use of dialog --infobox only shows static messages. Users cannot see the actual output of long-running commands like apt-get update or install, which may cause confusion if the operation takes time.

4. Basic Error Handling:

The script logs most errors but does not provide detailed or context-specific error messages to the user. If installation fails, the reason (e.g., incorrect package name, network issue) is not immediately visible in the dialog interface.

5. No GUI Fallback:

The script is entirely dialog-based and does not fall back to a command-line interface or graphical interface if dialog is missing. It assumes that dialog is installed and working properly.

6. Temporary Files Not Auto-Cleared:

Files created in /tmp/package_installer/ (like logs, process lists, package info) are not automatically deleted after script exit. This could consume disk space if the script is run frequently.

7. Limited Process Monitoring:

The script displays only the top 20 processes based on memory usage. It doesn’t offer dynamic or interactive process management features like top or htop.

8. No Package Search Functionality:

There is no option to search for packages by keyword; the user must know the exact package name for install, remove, or info commands.

9. Conclusion and Future Work

9.1 Summary of Accomplishments

The Package Installer Menu script successfully achieves its objective of simplifying Linux package management by offering a guided, interactive menu interface. It is particularly useful for beginners and system administrators who prefer a safer and more intuitive method of installing and managing packages.

Key accomplishments include:

1. Simplified Package Operations:

Users can install, remove, check, and inspect packages without needing to recall apt or dpkg commands, reducing complexity and errors.

2. Interactive Dialog-Based Interface:

The script utilizes the dialog toolkit to provide structured menus, input prompts, and informative message boxes that enhance usability and minimize terminal syntax issues.

3. Logging and Backup Mechanism:

All operations are logged persistently in /var/log/package_installer.log, and the script provides functionality to back up this log for administrative or troubleshooting purposes.

4. Modular Design:

Each major operation (installation, removal, info lookup, etc.) is implemented as a separate function, making the script clean, extensible, and easy to maintain.

5. System Insight Tools:

The addition of features to display currently logged-in users and top-running processes increases the script's utility as a lightweight system monitoring tool.

9.2 Learnings from the Project

This project helped develop and reinforce several practical skills in Linux scripting and software utility design:

Bash Scripting Proficiency:

The use of advanced shell scripting features such as conditional branching, functions, process substitution, and secure variable handling was essential throughout development.

User-Centric Design:

Creating a responsive and intuitive UI required iterating on layout design, input validation, and feedback mechanisms to ensure a smooth user experience.

System Command Integration:

Gaining hands-on experience with apt, dpkg, dialog, ps, and who commands improved overall Linux proficiency and scripting versatility.

Error Handling and Logging:

Incorporating real-time feedback, graceful fallbacks, and persistent logging helped make the tool robust and reliable under typical usage scenarios.

9.3 Future Enhancements

To further improve the functionality and user experience of the Package Installer Menu, the following enhancements are proposed:

Functional Additions

1. Cross-Distribution Support:

Extend compatibility to RPM-based systems by adding conditional logic for yum, dnf, or zypper package managers.

2. Dependency Tree Visualization:

Implement features to show dependency chains or suggest missing dependencies before installation.

3. Batch Install/Remove Mode:

Allow users to queue multiple packages for installation or removal in one go.

Usability and Reporting

1. Log Export Options:

Enable users to export logs to user-defined locations or email them for remote troubleshooting.

2. Progress Indicators:

Add progress bars or percentage indicators to long operations like updates or bulk installations.

3. Searchable Package Browser:

Integrate an interactive search feature to browse available packages using keywords or categories.

Advanced Features

1. Expert Mode Terminal:

Provide an embedded terminal or command execution option for advanced users who wish to run raw commands within the same UI.

2. Script Modularization:

Split the script into separate modular files (e.g., installer.sh, remover.sh, viewer.sh) to promote reusability and simplified development.

Educational Tools

1. Learning Mode:

Show explanations or command breakdowns during operations to help users understand what the script is doing behind the scenes.

10. References

Below are the key references, tools, and documentation used in the development of Package Installer Menu.

1.Linux Man Pages – Official documentation for commands:

2. Python GUI Libraries – Used for interface development:

3. Networking Guides – For troubleshooting methodologies:

- Linux Network Administrator's Guide (tldp.org/LDP/nag2/index.html)
- Ubuntu Networking Documentation (help.ubuntu.com/networking)

11. Appendices

11.1 Configuration Files

This script uses and interacts with the following configuration and system files:

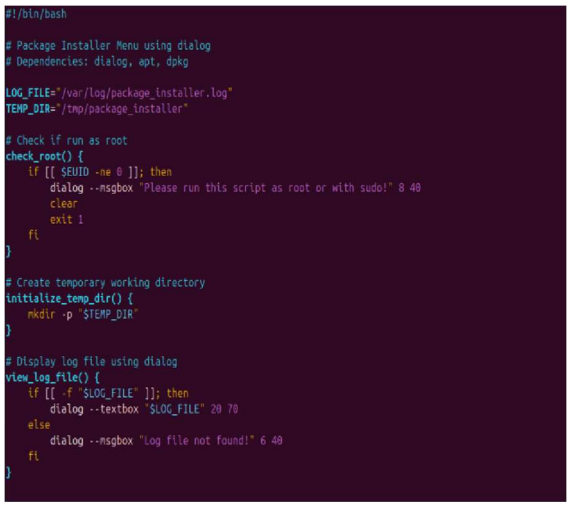
File/Directory	**Purpose**
/var/log/package_installer.log	Custom log file used to store installation and removal logs with timestamps
/tmp/package_installer/	Temporary working directory for logs, backups, and package markers
/etc/apt/	Contains APT configuration and repository information (used indirectly)
/usr/bin/apt, /usr/bin/dpkg	Executables used to perform package operations

Temporary files used internally:

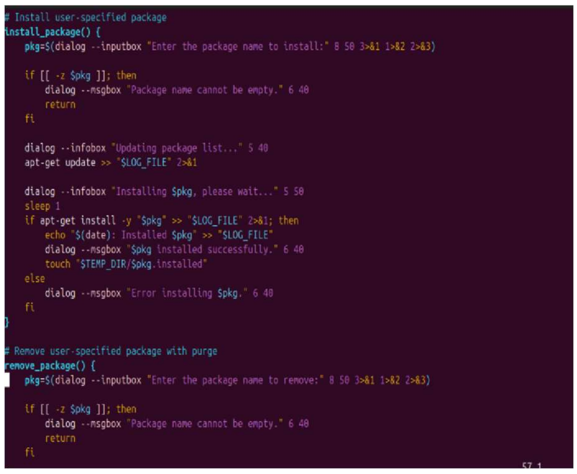
- /tmp/package_installer/pkg_info.txt – holds package metadata temporarily for viewing.
- /tmp/package_installer/users.txt – stores output from the who command.
- /tmp/package_installer/processes.txt – stores output from ps command for display.
- /tmp/package_installer/backup.log – backup copy of the main log file.
- /tmp/package_installer/[pkgname].installed – marker files indicating installed packages.

11.2 Script Listings

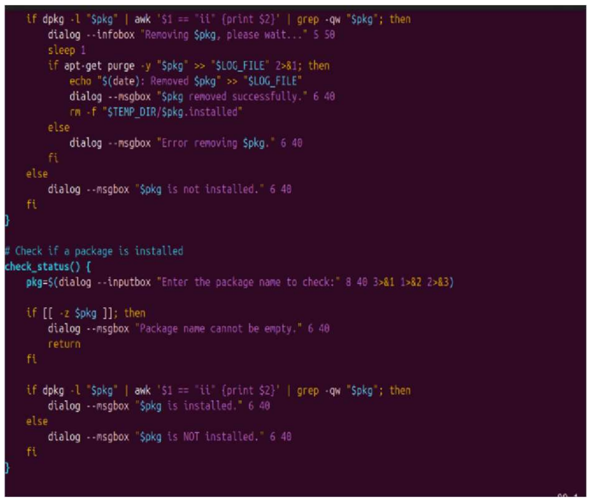
Below is the complete Bash script implementing the Package Installer Menu:



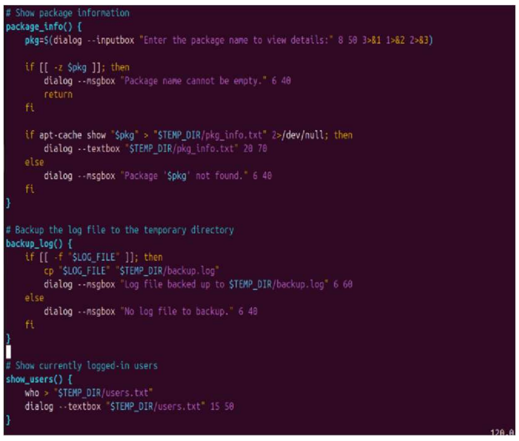
```bash
#!/bin/bash

# Package Installer Menu using dialog
# Dependencies: dialog, apt, dpkg

LOG_FILE="/var/log/package_installer.log"
TEMP_DIR="/tmp/package_installer"

# Check if run as root
check_root() {
    if [[ $EUID -ne 0 ]]; then
        dialog --msgbox "Please run this script as root or with sudo!" 8 40
        clear
        exit 1
    fi
}

# Create temporary working directory
initialize_temp_dir() {
    mkdir -p "$TEMP_DIR"
}

# Display log file using dialog
view_log_file() {
    if [[ -f "$LOG_FILE" ]]; then
        dialog --textbox "$LOG_FILE" 20 70
    else
        dialog --msgbox "Log file not found!" 6 40
    fi
}
```



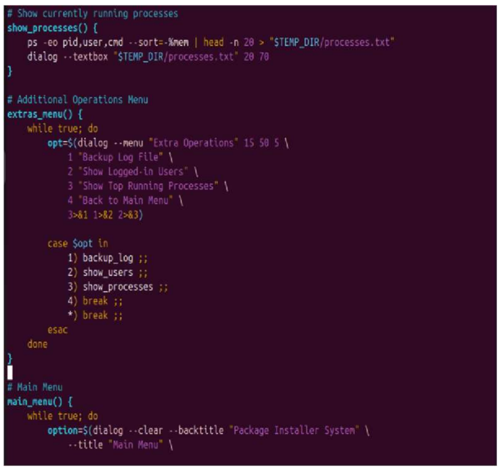
```bash
# Install user-specified package
install_package() {
    pkg=$(dialog --inputbox "Enter the package name to install:" 8 50 3>&1 1>&2 2>&3)

    if [[ -z $pkg ]]; then
        dialog --msgbox "Package name cannot be empty." 6 40
        return
    fi

    dialog --infobox "Updating package list..." 5 40
    apt-get update >> "$LOG_FILE" 2>&1

    dialog --infobox "Installing $pkg, please wait..." 5 50
    sleep 1
    if apt-get install -y "$pkg" >> "$LOG_FILE" 2>&1; then
        echo "$(date): Installed $pkg" >> "$LOG_FILE"
        dialog --msgbox "$pkg installed successfully." 6 40
        touch "$TEMP_DIR/$pkg.installed"
    else
        dialog --msgbox "Error installing $pkg." 6 40
    fi
}

# Remove user-specified package with purge
remove_package() {
    pkg=$(dialog --inputbox "Enter the package name to remove:" 8 50 3>&1 1>&2 2>&3)

    if [[ -z $pkg ]]; then
        dialog --msgbox "Package name cannot be empty." 6 40
        return
    fi
```



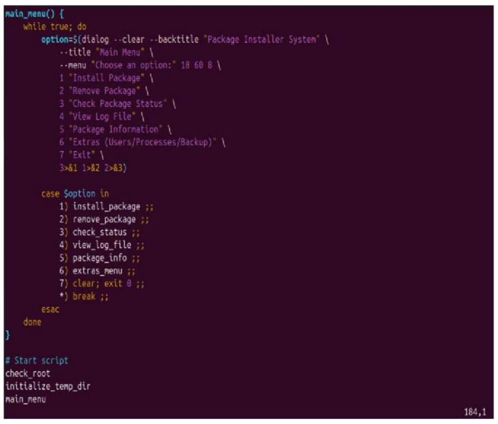
```bash
    if dpkg -l "$pkg" | awk '$1 == "ii" {print $2}' | grep -qw "$pkg"; then
        dialog --infobox "Removing $pkg, please wait..." 5 50
        sleep 1
        if apt-get purge -y "$pkg" >> "$LOG_FILE" 2>&1; then
            echo "$(date): Removed $pkg" >> "$LOG_FILE"
            dialog --msgbox "$pkg removed successfully." 6 40
            rm -f "$TEMP_DIR/$pkg.installed"
        else
            dialog --msgbox "Error removing $pkg." 6 40
        fi
    else
        dialog --msgbox "$pkg is not installed." 6 40
    fi
}

# Check if a package is installed
check_status() {
    pkg=$(dialog --inputbox "Enter the package name to check:" 8 40 3>&1 1>&2 2>&3)

    if [[ -z $pkg ]]; then
        dialog --msgbox "Package name cannot be empty." 6 40
        return
    fi

    if dpkg -l "$pkg" | awk '$1 == "ii" {print $2}' | grep -qw "$pkg"; then
        dialog --msgbox "$pkg is installed." 6 40
    else
        dialog --msgbox "$pkg is NOT installed." 6 40
    fi
}
```



```
# Show package information
package_info() {
    pkg=$(dialog --inputbox "Enter the package name to view details:" 8 50 3>&1 1>&2 2>&3)

    if [[ -z $pkg ]]; then
        dialog --msgbox "Package name cannot be empty." 6 40
        return
    fi

    if apt-cache show "$pkg" > "$TEMP_DIR/pkg_info.txt" 2>/dev/null; then
        dialog --textbox "$TEMP_DIR/pkg_info.txt" 20 70
    else
        dialog --msgbox "Package '$pkg' not found." 6 40
    fi
}

# Backup the log file to the temporary directory
backup_log() {
    if [[ -f "$LOG_FILE" ]]; then
        cp "$LOG_FILE" "$TEMP_DIR/backup.log"
        dialog --msgbox "Log file backed up to $TEMP_DIR/backup.log" 6 60
    else
        dialog --msgbox "No log file to backup." 6 40
    fi
}

# Show currently logged-in users
show_users() {
    who > "$TEMP_DIR/users.txt"
    dialog --textbox "$TEMP_DIR/users.txt" 15 50
}
                                                                     120,0
```

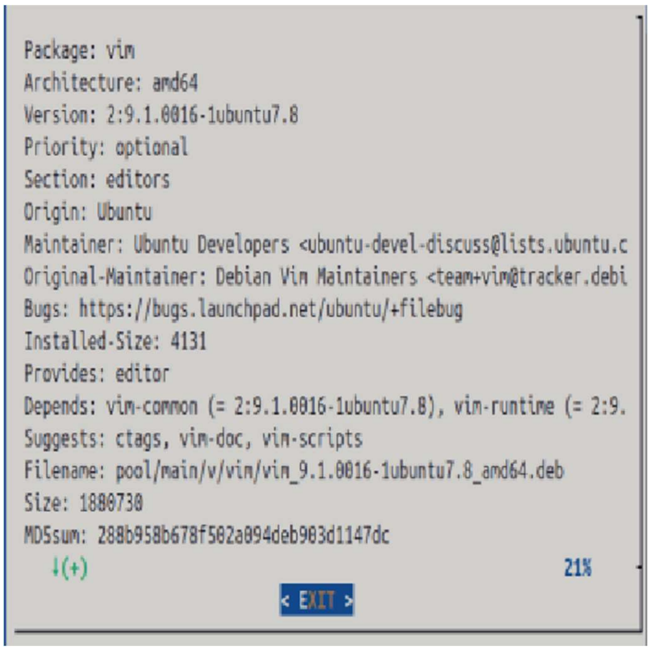


```
# Show currently running processes
show_processes() {
    ps -eo pid,user,cmd --sort=-%mem | head -n 20 > "$TEMP_DIR/processes.txt"
    dialog --textbox "$TEMP_DIR/processes.txt" 20 70
}

# Additional Operations Menu
extras_menu() {
    while true; do
        opt=$(dialog --menu "Extra Operations" 15 50 5 \
            1 "Backup Log File" \
            2 "Show Logged-in Users" \
            3 "Show Top Running Processes" \
            4 "Back to Main Menu" \
            3>&1 1>&2 2>&3)

        case $opt in
            1) backup_log ;;
            2) show_users ;;
            3) show_processes ;;
            4) break ;;
            *) break ;;
        esac
    done
}

# Main Menu
main_menu() {
    while true; do
        option=$(dialog --clear --backtitle "Package Installer System" \
            --title "Main Menu" \
```

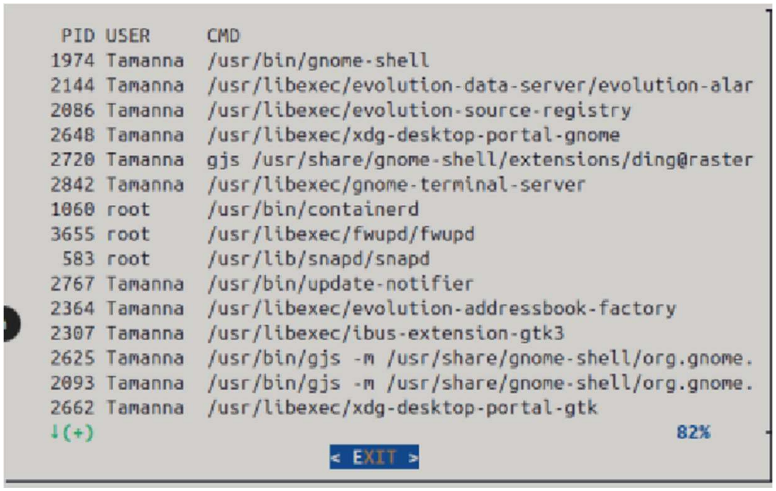


```
main_menu() {
    while true; do
        option=$(dialog --clear --backtitle "Package Installer System" \
            --title "Main Menu" \
            --menu "Choose an option:" 18 60 8 \
            1 "Install Package" \
            2 "Remove Package" \
            3 "Check Package Status" \
            4 "View Log File" \
            5 "Package Information" \
            6 "Extras (Users/Processes/Backup)" \
            7 "Exit" \
            3>&1 1>&2 2>&3)

        case $option in
            1) install_package ;;
            2) remove_package ;;
            3) check_status ;;
            4) view_log_file ;;
            5) package_info ;;
            6) extras_menu ;;
            7) clear; exit 0 ;;
            *) break ;;
        esac
    done
}

# Start script
check_root
initialize_temp_dir
main_menu
                                                                     184,1
```

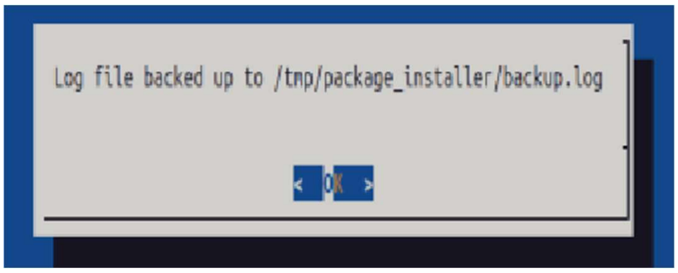
11.3 Additional Screenshots or Data



```
Package: vim
Architecture: amd64
Version: 2:9.1.0016-1ubuntu7.8
Priority: optional
Section: editors
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.c
Original-Maintainer: Debian Vim Maintainers <team+vim@tracker.debi
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 4131
Provides: editor
Depends: vim-common (= 2:9.1.0016-1ubuntu7.8), vim-runtime (= 2:9.
Suggests: ctags, vim-doc, vim-scripts
Filename: pool/main/v/vim/vim_9.1.0016-1ubuntu7.8_amd64.deb
Size: 1880730
MD5sum: 288b958b678f502a094deb903d1147dc
  ↓(+)                                                     21%
```



```
  PID USER     CMD
 1974 Tamanna  /usr/bin/gnome-shell
 2144 Tamanna  /usr/libexec/evolution-data-server/evolution-alar
 2086 Tamanna  /usr/libexec/evolution-source-registry
 2648 Tamanna  /usr/libexec/xdg-desktop-portal-gnome
 2720 Tamanna  gjs /usr/share/gnome-shell/extensions/ding@raster
 2842 Tamanna  /usr/libexec/gnome-terminal-server
 1060 root     /usr/bin/containerd
 3655 root     /usr/libexec/fwupd/fwupd
  583 root     /usr/lib/snapd/snapd
 2767 Tamanna  /usr/bin/update-notifier
 2364 Tamanna  /usr/libexec/evolution-addressbook-factory
 2307 Tamanna  /usr/libexec/ibus-extension-gtk3
 2625 Tamanna  /usr/bin/gjs -m /usr/share/gnome-shell/org.gnome.
 2093 Tamanna  /usr/bin/gjs -m /usr/share/gnome-shell/org.gnome.
 2662 Tamanna  /usr/libexec/xdg-desktop-portal-gtk
↓(+)                                                     82%
```



```
Log file backed up to /tmp/package_installer/backup.log
```