Project Based Evaluation

Project Report

Semester-IV (Batch-2023)

AutoStash – Smart Linux Backup System

Supervised by:

Ms. Parul

Submitted by:

Abhinav Rathee, 2310991675 (G19)
Akanksha Mishra, 2310991681 (G19)
Anuj Kumar, 2310991699 (G19)
Anushi, 2310991700 (G19)

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

Abstract

In Linux systems, managing backups is essential but often ignored due to the complexity involved in configuring them manually through command-line tools. This project, **AutoStash**, is a smart, automated, and user-friendly backup system designed specifically for Linux environments. It allows users to back up important files or even system configurations like /etc, automatically push backups to GitHub repositories using a securely saved token (without terminal interaction), schedule regular backups via cron or systemd, and optionally encrypt and compress backups.

AutoStash also includes a graphical user interface (GUI) built using Python's Tkinter, which enables folder selection, repository management, backup monitoring, and restoration with just a few clicks. System resources such as CPU, RAM, and disk usage are monitored during backup operations, and detailed logs are stored in /var/log for transparency. This project integrates fundamental concepts of system scripting, automation, encryption, and version control, packaged in a modular and maintainable architecture.

This project not only showcases automation and scripting in Linux but also emphasizes real-world system maintenance practices like version control, scheduling, logging, and encryption. By combining these features into a single, user-friendly tool, AutoStash serves as both a practical utility for daily use and a learning resource for students and developers aiming to deepen their understanding of Linux-based automation and Python development.

Table of Contents

S.No.	Content	Page No.
1.	Introduction: • Background and Motivation • Objectives of the Project • Scope of the Work • Report Structure	4-5
2.	System Environment: • Hardware and Software Requirements • Linux Distribution and Version • Tools and Utilities Used	6
3.	Conceptual Overview: • Key Concepts Related to the Project • Relevant System Components and Files • Linux Commands and Services Involved	7-8
4.	UML Diagrams: • Use Case Diagram • Activity Diagram • Sequence Diagram • Class Diagram (if applicable)	9-10
5.	Implementation Details: • Step-by-Step Configuration/Development • Commands and Scripts Used • Screenshots and Outputs	11-15

6.	Security and Optimization: • Hardening Measures Taken • Performance Tuning and Efficiency • Backup and Recovery Measures	16
7.	Testing and Validation: • Test Scenarios and Expected Results • Troubleshooting Techniques • Logs and Monitoring Tools	16-17
8.	Challenges and Limitations: • Problems Faced During Implementation • Workarounds and Fixes • Known Issues or Constraints	17
9.	Conclusion and Future Work: • Summary of Accomplishments • Learnings from the Project • Future Enhancements	17-18
10.	References	19
11.	Appendices: • Configuration Files • Additional Screenshots or Data	20-21

1. Introduction:

1.1 Background and Motivation:

Data loss due to system failure, misconfiguration, or accidental deletion is a common issue in Linux environments. Backing up files and system settings can prevent these problems, but many users and admins avoid doing so due to the technical nature of backup tools and scripts. This project bridges that gap by offering a visual and automated backup solution.

1.2 Objectives of the Project:

- Develop an intuitive GUI-based backup tool for Linux
- Allow users to select folders and back them up to GitHub
- Securely store GitHub tokens and push backups without CLI
- Offer optional encryption and compression
- Schedule backups using cron or systemd timers
- Monitor system resources during backups
- Maintain logs in /var/log/ for auditing and troubleshooting

1.3 Scope of the Work:

This project focuses on automated and secure backups of user-selected folders and Linux system files in a visual, scheduled, and monitored way. It is not a full system snapshot tool like Timeshift but is ideal for user files, configurations, and lightweight system backups. The system is focused on Linux-based systems only and handles folder-level backups (not entire disk imaging). It supports encryption, compression, scheduled execution, and critical configuration file handling (/etc).

1.4 Report Structure:

This report is divided into sections covering the system environment, conceptual understanding, UML diagrams, implementation, security and testing, along with project learnings and future scope.

2. System Environment:

2.1 Hardware and Software Requirements:

Backing up important data in Linux typically requires manual scripts, terminal commands, and careful configuration. This can be overwhelming for non-experts. Additionally, common issues like forgetting to back up, improperly storing tokens, or overloading the system during backup reduce the effectiveness of current solutions. AutoStash solves this with automation, GUI control, encryption, and monitoring — all in one tool.

Software:

- Python 3.8+
- Linux OS (Ubuntu, Fedora, RHEL, Kali)
- Git & GitHub Account (for backup repository)
- Tkinter (GUI)
- cron or systemd for scheduling
- GPG for optional encryption

Hardware:

- 2 GB RAM minimum
- At least 200 MB disk space for logs and temporary files
- Internet connection for pushing backups to GitHub

2.2 Linux Distribution and Version:

Tested on Ubuntu 24.04, Fedora 39, RHEL 9

2.3 Tools and Utilities Used:

- Tkinter for GUI, gnupg for encryption
- cron or systemd for scheduling
- psutil for monitoring, GitPython for git integration
- GPG for optional encryption

3. Conceptual Overview:

3.1 Key Concepts Related to the Project:

- Incremental vs Full Backups (controlled via flags)
- GPG Encryption for securing archives
- Git-based remote repositories for version control
- System resource monitoring using multithreading
- GUI Layer (main.py): Provides the user interface using Tkinter
- Backup Logic: Handles compression, encryption, and file archiving
- GitHub Integration: Uses keyring to store tokens securely, supports auto-push
- Scheduler: Uses cron/systemd for timed backups
- Monitoring: Tracks CPU, memory, disk during backup
- Restoration: Provides visual timeline to restore old backups

3.2 Relevant System Components and Files:

- /etc/ → system config files
- ~/.autostash/ → user config and tokens
- /var/log/autostash/ → backup logs

Some features and functionalities:

- **Folder Selection with GUI:** Select multiple folders using a modern GUI
- **Silent GitHub Integration:** Save GitHub token securely; push backups silently
- **Backup of System Files (/etc):** Allows protected system configuration files to be backed up

- **Scheduling via Cron or Systemd:** Set backups to run automatically at chosen intervals
- **Log Storage:** Keeps backup history and logs in /var/log/autostash
- **Optional GPG Encryption:** Encrypt backup files before uploading
- **Compression:** Reduce backup file size with compression
- **Resource Monitoring**: Real-time display of CPU, RAM, disk usage
- **GUI Timeline & Restoration:** Visual history with time-stamped backups and restore options
- **Advanced Options:** Export/import settings, manage GPG keys, verify backups.

3.3 Linux Commands and Services Involved:

- git clone/push, gpg, tar, rsync
- cron, systemctl for scheduling
- sudo, chmod, keyring for permissions

Tools and technology used:

Category	Tools/Tech Used
Programming	Python (v3.8+)
GUI	Tkinter
Scheduling	cron / systemd
Encryption	GPG (`python-gnupg`)
Git Integration	Git CLI + `keyring`
Monitoring	psutil / resource APIs
Version Control	GitHub
File Storage	`.autostash` in home directory
Logging	`/var/log/autostash/`

Figure 3.5.1 Lists Tools Used to Build AutoStash

4. UML Diagram:

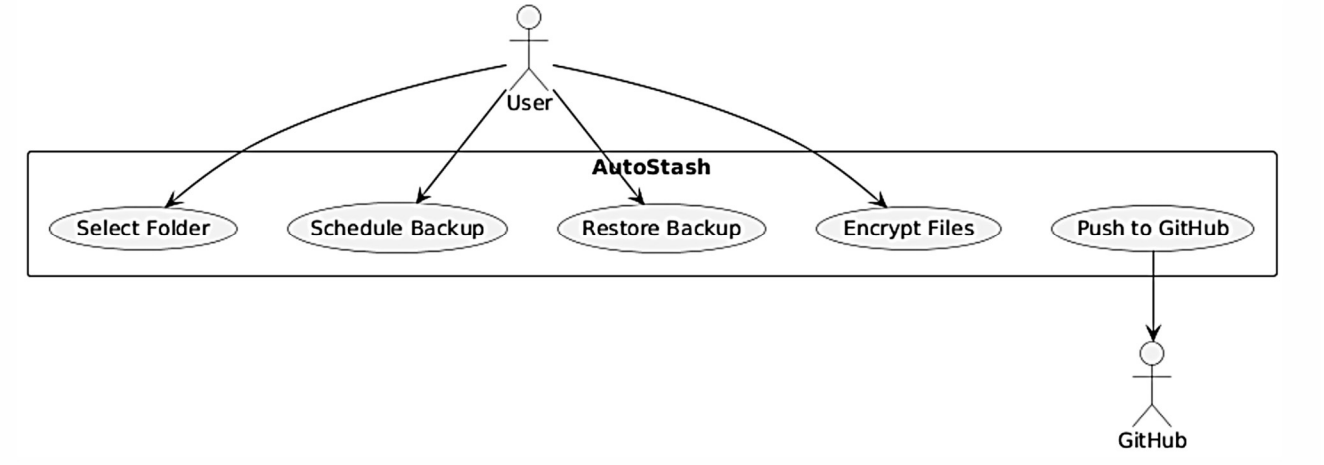

Figure 4.1 Use Case Diagram of AutoStash

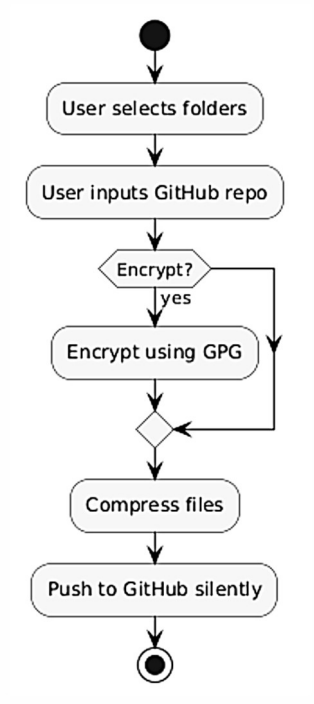

Figure 4.2 Activity Diagram of AutoStash

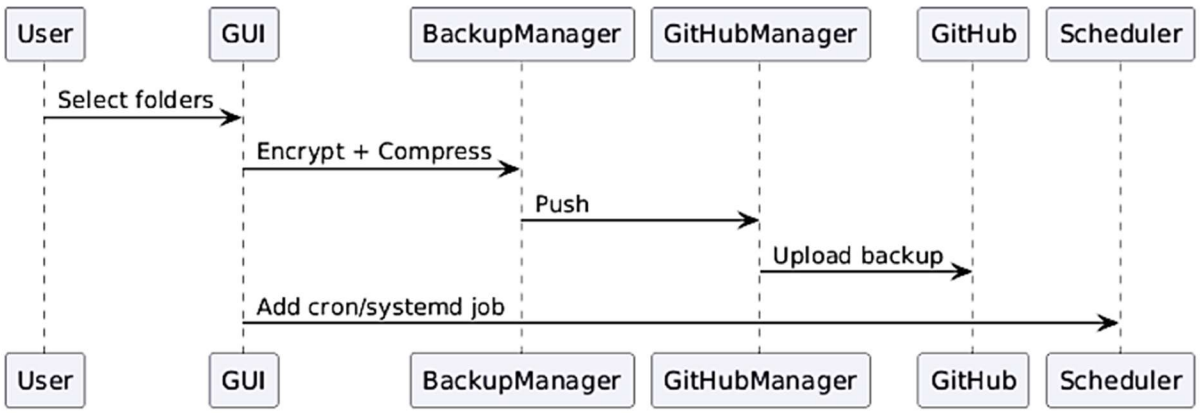

Figure 4.3 Sequence Diagram of AutoStash

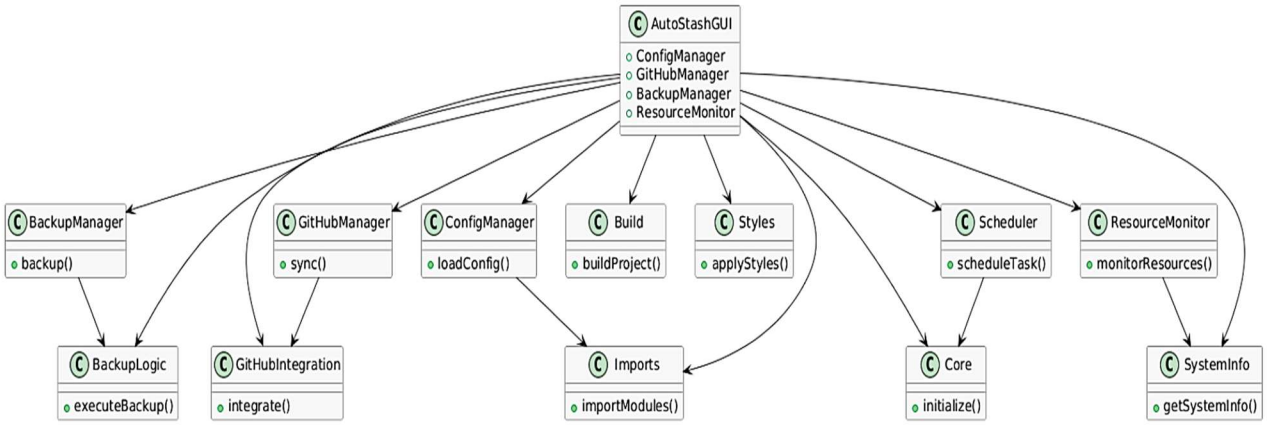

Figure 4.4 Class Diagram of AutoStash

5. Implementation Detail:

5.1 Step-by-Step Configuration/Development:

- Token stored using keyring.set_password("autostash", "github_token", token)
- Backup executed via BackupManager().run(...)
- System resource monitored via ResourceMonitor.start_monitoring()
- Scheduler starts background thread using threading.Thread(target=...)

5.2 Commands and Scripts Used:

- rsync, tar, gpg, git push, cron, systemctl
- Python modules: gnupg, tkinter, logging, subprocess

5.3 Screenshots and Outputs:

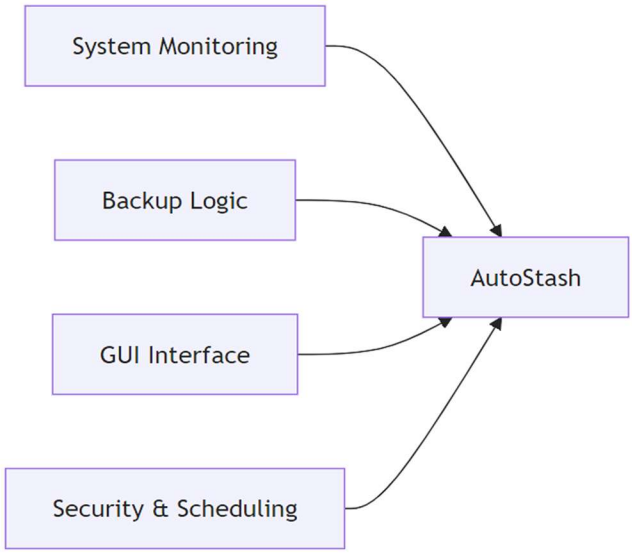

*Figure **5.3.1** Modules - AutoStash is made up of*

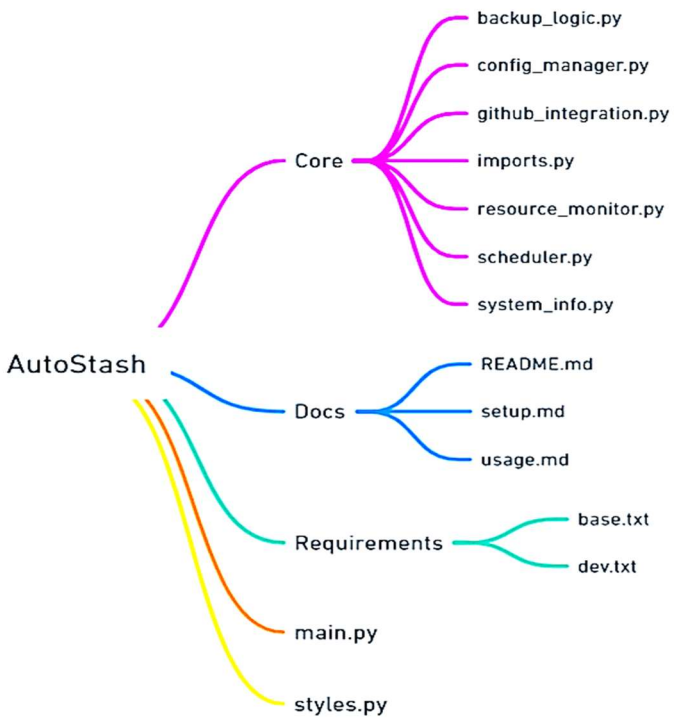

Figure 5.3.2 Shows the Folder Structure of AutoStash Project

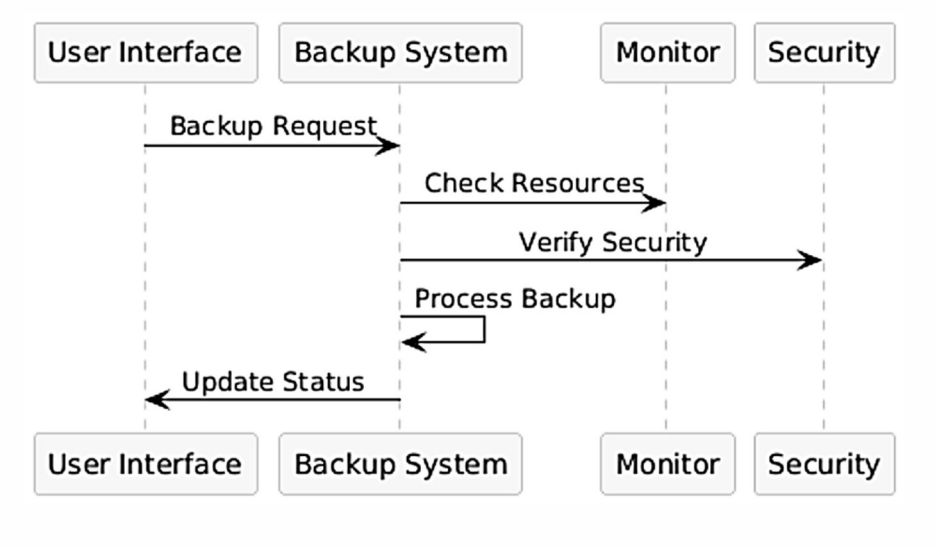

*Figure **5.3.3** Shows the System Architecture of AutoStash*

Screenshots of the GUI:

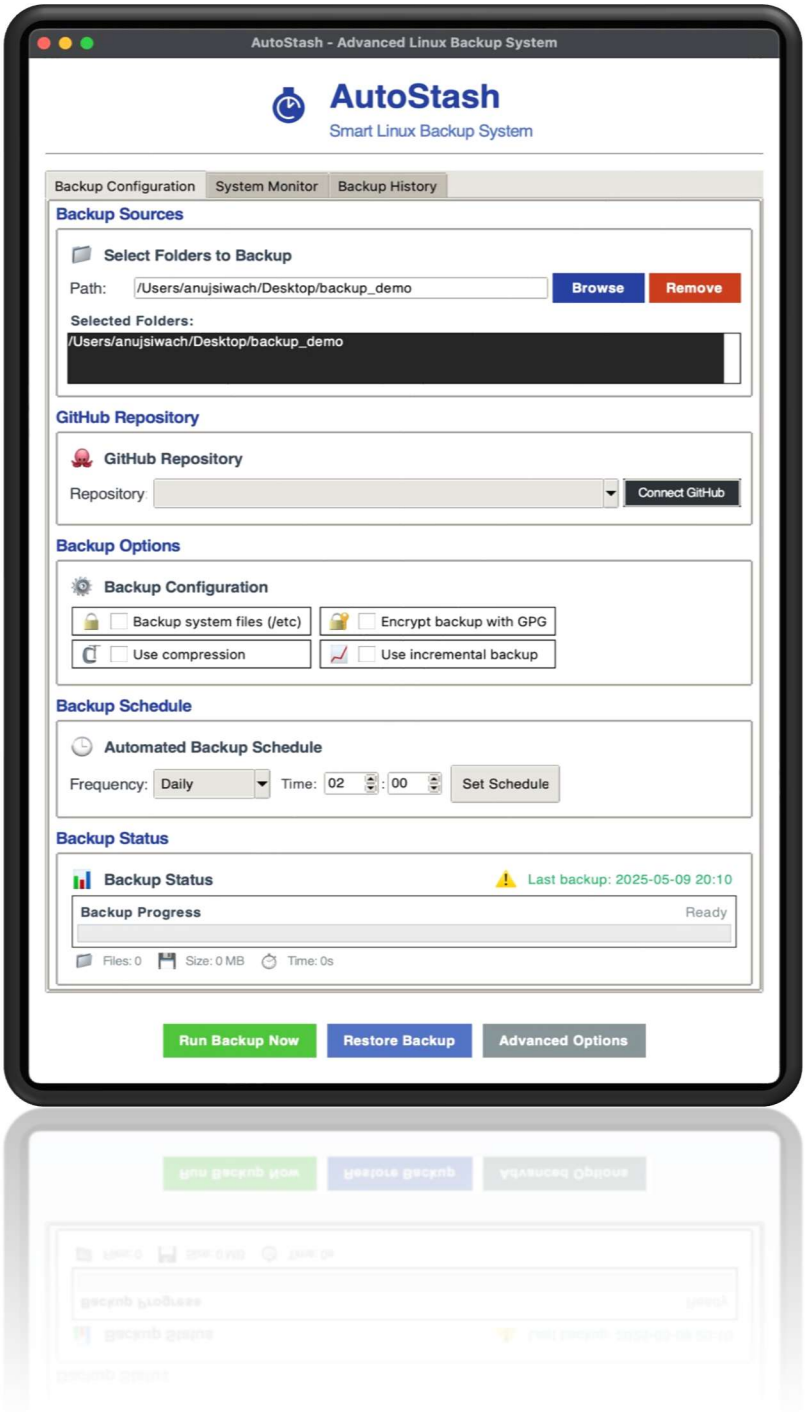

Figure 5.3.4 Main Landing Page of AutoStash – Backup Configurations

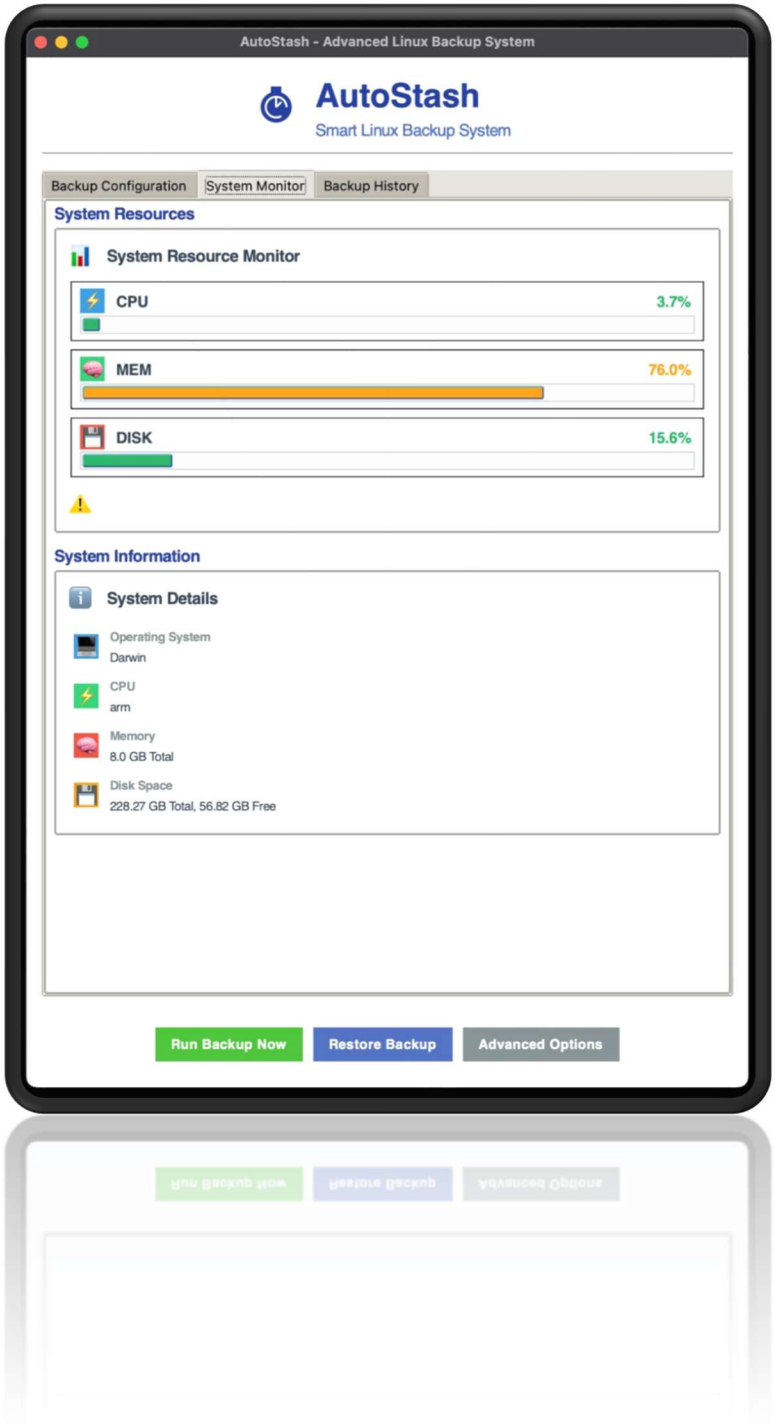

Figure 5.3.5 System Monitoring under AutoStash While Backing up

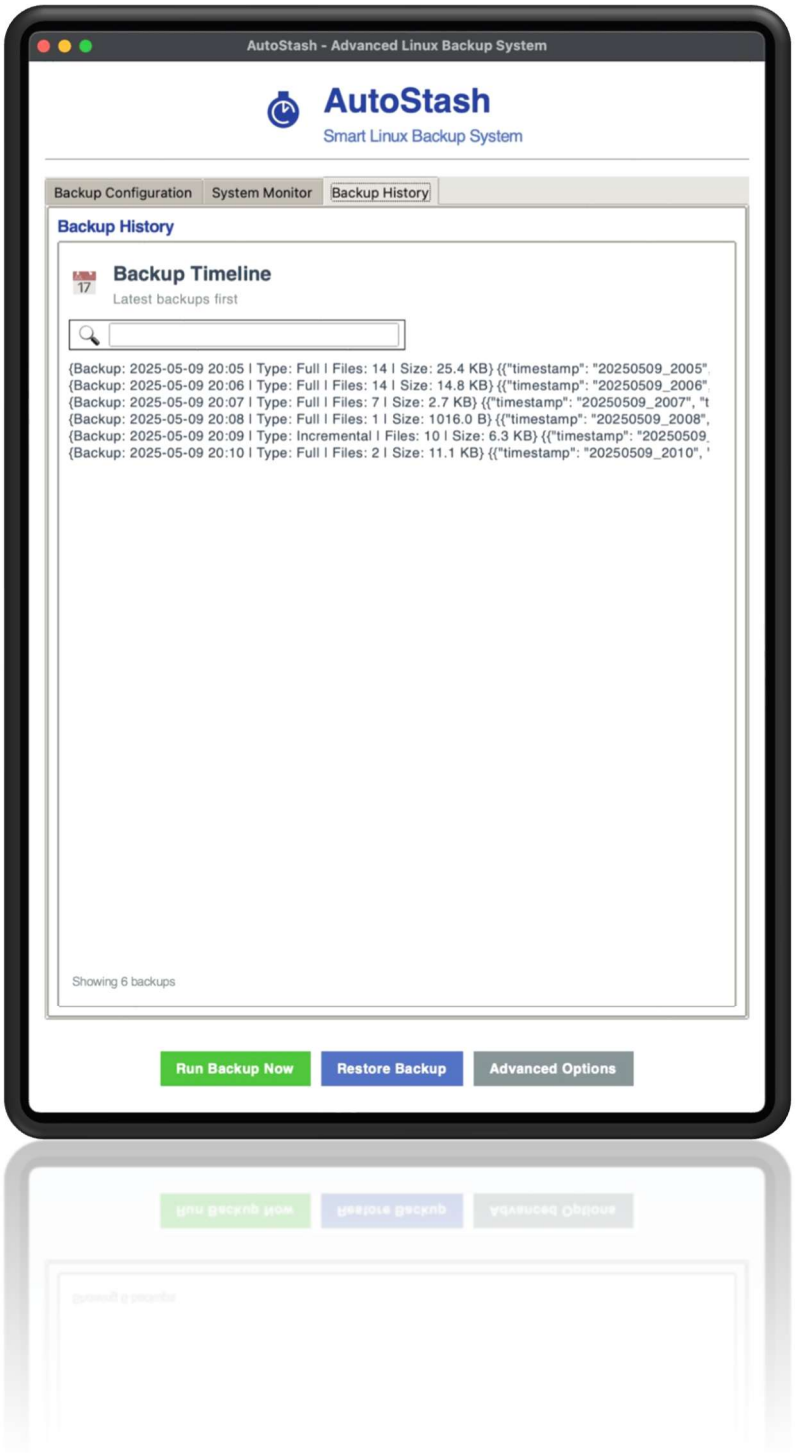

Figure 5.3.6 Backup History and Timeline

6. Security and Optimization:

6.1 Hardening Measures Taken:

- GitHub token encrypted and saved via keyring
- GPG key verification
- File permission checks with os.access()

6.2 Performance Tuning and Efficiency:

- rsync used for incremental backup
- GUI remains responsive via background threads
- Logs filtered for only relevant info using logging levels

6.3 Backup and Recovery Measures:

- Metadata stored in metadata.json
- /var/log/autostash/backup.log maintains logs
- restore() function reads latest valid backup folder from repo

7. Testing and Validation:

7.1 Test Scenarios and Expected Results:

- Invalid token → Error prompt
- Folder with no read permission → Exception raised
- No internet during push → Git error logged

7.2 Troubleshooting Techniques:

- Refer to backup.log and error.log
- Use verify_backups() to test file integrity

7.3 Logs and Monitoring Tools:

- Logs include INFO, WARNING, ERROR tags
- psutil used to monitor real-time CPU, memory, disk

8. Challenges and Limitations:

8.1 Problems Faced During Implementation:

- GUI hanging on long backups (fixed via threading)
- GPG missing on some systems (handled via error prompt)

8.2 Workarounds and Fixes:

- Used subprocess.run() instead of shell commands
- Implemented error dialogs with tkinter.messagebox

8.3 Known Issues or Constraints:

- Not a full system image backup
- Cannot restore without internet (GitHub dependency)
- Can be improved to a much greater scale using cloud

9. Conclusion and Future Work:

9.1 Summary of Accomplishments:

- Successfully created automated backup system with restore capability
- Built cross-module interaction using OOP and callbacks

9.2 Learnings from the Project:

- Applied threading, subprocess handling, encryption, and Git operations
- Understood Linux environment paths and security concerns

9.3 Future Enhancements:

- Add support for cloud providers (Google Drive, Dropbox)
- Build web GUI using Flask
- Add notification service via email/Discord webhook

10. References:

- Python Official Documentation
  - Retrieved from https://docs.python.org/3
  - Used for understanding standard libraries, GUI (Tkinter), subprocess handling, threading, and file I/O operations in Python.
- GitHub Developer Docs
  - GitHub, Inc. (n.d.). GitHub REST API Documentation. Retrieved from https://docs.github.com
  - Used for automating repository interactions, managing personal access tokens, and understanding version control processes.
- GeeksforGeeks – Linux Shell Scripting
  - GeeksforGeeks. (n.d.). Linux Shell Scripting and System Administration Tutorials. Retrieved from https://www.geeksforgeeks.org
  - Helped understand shell command automation and integrating system tools via Python subprocesses.
- TutorialsPoint – cron, GPG, and system administration
  - TutorialsPoint. (n.d.). Linux Administration & Scheduling Tasks Using cron/systemd. Retrieved from https://www.tutorialspoint.com
  - Used to implement scheduling for periodic backups and setting up encryption using GPG.
- Stack Overflow Discussions on Tkinter & GitHub integration
  - Stack Exchange Inc. (n.d.). Stack Overflow Developer Community. Retrieved from https://stackoverflow.com
  - Consulted for troubleshooting GUI issues (Tkinter), error handling, and securely pushing commits to GitHub from Python.
- GitHub – Version Control and Collaboration
  - GitHub. (n.d.). About GitHub: Version Control for Projects. Retrieved from https://github.com/
  - Core platform used for storing, syncing, and versioning backup files automatically from the application.

11. Appendices:

11.1 Configuration Files:

- ~/.autostash/config.json
- /var/log/autostash/backup.log

11.2 Script Listings:

- backup_logic.py: Core backup handling
- scheduler.py: Scheduler engine
- resource_monitor.py: Real-time monitor
- system_info.py: System metadata gathering

11.3 Additional Screenshots or Data:

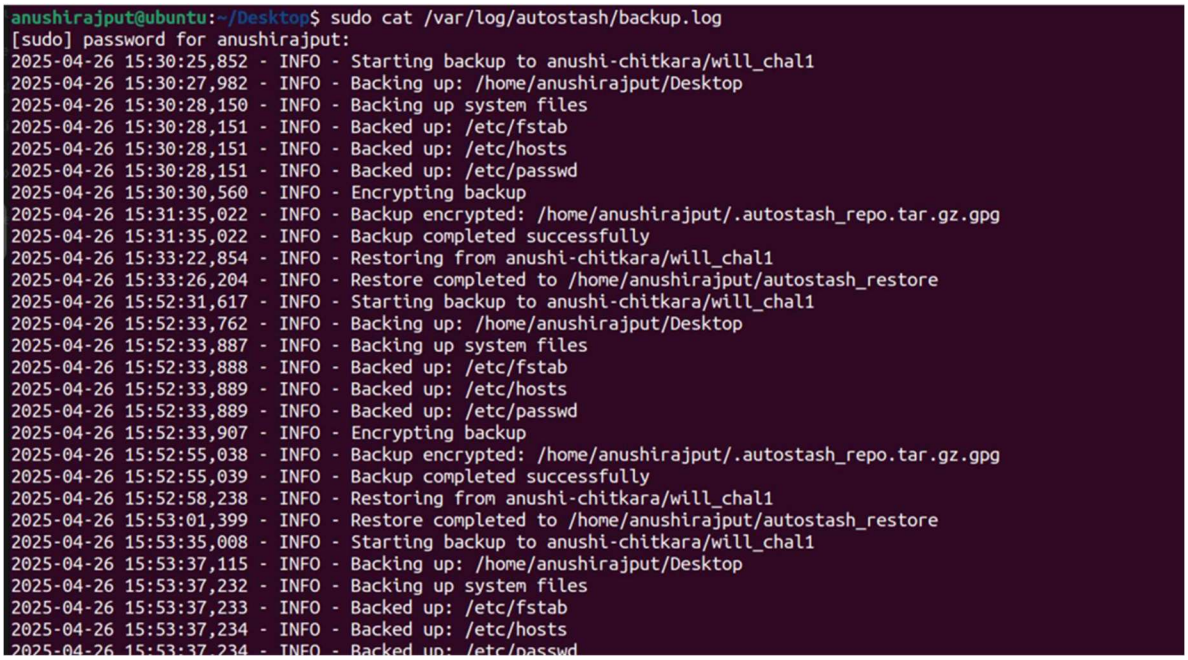

Figure 11.3.1 Save Backup Logs in /var/log/

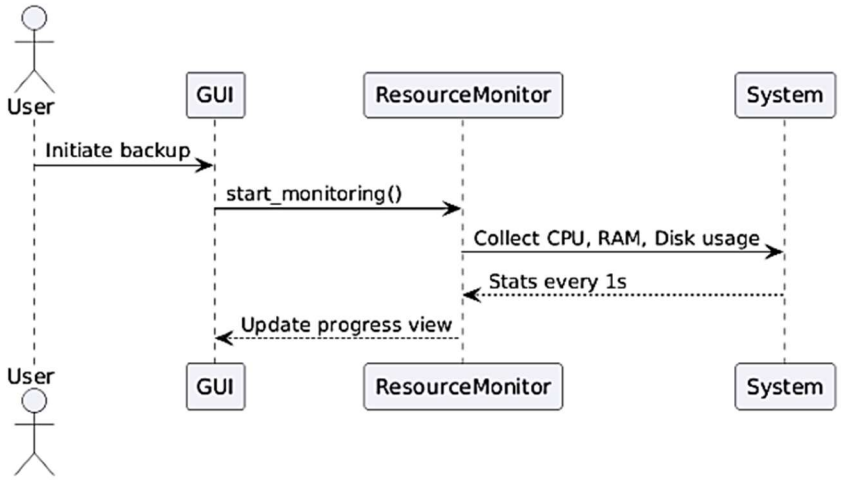

Figure 11.3.2 Interaction between the GUI and the system resource monitor during a backup

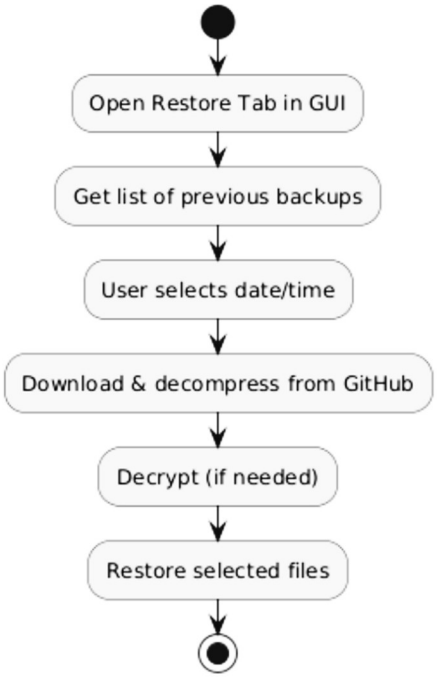

Figure 11.3.3 Timeline and process flow during restoration of a previous backup