DEPARTMENT OF COMPUTER SCIENCE

Module Code: COM00193M

Submitted in part fulfilment of the degree of MSc in Cyber Security.

Google Drive Sync Artifact Parser for Autopsy

Keshav Joshi

Supervisor: Angus Marshall

Date: 4th September 2025

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Angus Marshall, for his guidance, support, and invaluable feedback throughout this project. His expertise and encouragement were fundamental to the successful completion of thi research.

I am also grateful to my family and friends for their encouragement during my studies. Their support has been a source of motivation throughout this journey.

I would also like to extend my appreciations to my colleagues, peers and all those who directly or indirectly contributed to this work, offering advice, assistance and inspiration along the way.

Finally, I would like to thank the University of York for providing the resources and academic environment that enabled this research.

Statement of Ethics

This research project was carried out in accordance with the ethical guidelines of the University of York's Department of Computer Science.

An ethics checklist was completed and approved by the Professional Services Research Ethics Committee (PSEC). No personal or sensitive data was involved in this project. All experimental data was generated by ourselves within controlled test environments, using dummy accounts and virtual environments machines.

The Google Drive Terms of Service were followed, as only client-side artifacts created on the researcher’s test machines were analysed. No server-side or third-party user data was accessed, ensuring full compliance with ethical, legal, social, and professional standards.

Abstract

This project examines the forensic significance of contemporary Google Drive for Desktop (DriveFS) artifacts and fills a gap in open-source forensic tooling by developing a custom Autopsy ingest module.

Legacy artifacts (snapshot.db, sync_log.log, and sync_config.db, often referenced in earlier studies, are no longer present in recent DriveFS versions. Instead, forensic evidence is now stored in databases such as metadata_sqlite.db and mirror_metadata.db, as well as structured logs like structured_log_global and drive_fs.txt, and configuration files including LocalPrefs.json and startup_trace.json.

The project followed the Design Science Research methodology, integrating controlled experiments in virtual settings with iterative plugin creation and validation. It involved simulated file operations, including creation, renaming, deletion, and restoration. These artifacts were captured, reverse-engineered, and incorporated into an Autopsy parser built with Python.

Evaluation demonstrated that the module effectively extracts evidence, validated through manual queries, and cuts investigation time by over 90%. Its strengths are automation, forensic integrity, and extensibility. However, it has limitations such as sensitivity to software versions and no support for macOS.

This work contributes to both academic research and practical digital forensics by connecting research with operational requirements. It shows that open-source forensic platforms such as Autopsy can be expanded to include modern cloud artifacts, providing investigators with accessible solutions.

This project develops a custom Autopsy ingest module for parsing modern Google DriveFS artifacts, addressing the forensic gap left by obsolete legacy artifacts.

Table of Contents

ACKNOWLEDGEMENTS ii
Statement of Ethics iii
Abstract iv
Chapter 1: Introduction 1
1.1 Motivation and Aim 4
1.2 Research Questions 4
Chapter 2: Literature Review 5
2.1 Introduction 5
2.2 Prior Work on Cloud Client artifacts 6
2.3 Google Drive Artifact Analysis in Existing Research 8
2.4 Modernisation of Google Drive Client Artifacts in DriveFS 10
2.5 Tools for Artifact Analysis and Current Gaps 13
2.6 Summary of findings and research gap 15
Chapter 3: Project Plan 17
3.1 Introduction 17
3.2 Project stages and Activities 17
3.3 Tools and Environment 19
3.4 Research Question Alignment 19
3.5 Challenges and Backup Strategies 20
3.6 Ethical Considerations 20
Chapter 4: Implementation and Results 21
4.1 Introduction 21
4.2 Development Environment and Toolchain 21
4.3 Artifact Generation and Evidence Collection 24
4.4 Manual Artifact Analysis 27
4.5 Plugin Design and Implementation 28
4.5.0 Preliminaries: Autopsy Plugin Setup 28
4.5.1 Architecture 28
4.5.2 Artifact Mapping 30
4.5.3 Cross-Validation with Manual Analysis 32

4.5.4 Performance Metrics 35
4.5.5 Summary of Findings 36
4.6 Evaluation 38
4.6.1 Alignment with Research Questions 38
4.6.2 Forensic Utility and Workflow Integration 39
4.6.3 Strengths of the Developed Plugin 40
4.6.4 Limitations and Challenges 40
4.6.5 Contribution to Digital Forensic Practice 41
4.7 Summary 42
Chapter 5: Evaluation, Discussion and Conclusion 43
5.1 Introduction 43
5.2 Research Questions Evaluation 43
5.2.1 Wider Implications and Future Extensions 45
5.3 Reflection on Methodology 46
5.4 Alignment with Literature 47
5.5 Future Work 48
5.6 Conclusion 49
References 51
Appendices 53
Appendix A – Supplementary screenshots 53
A.1 Appendix A figure list: 53
A.2 Supporting Screenshots 55
A.3 Reproducibility Checklist 75
Appendix B – Google DriveFS Autopsy Plugin Source Code 76
B.1 Module File Location 76
B.2 Source Code (GoogleDriveIngestModule.py) 77

Table of Figures

Figure 1.1: Evolution of Google Drive Client Artifacts 2

Figure 2.1: Evolution of Google Drive client artifacts from legacy File Stream to modern DriveFS 13

Figure 3.1: Project stages (DSR framework) 18

Figure 4.1: Virtual machine creation in VMware Workstation 22

Figure 4.2: Login of test Google Drive account using dummy credentials 22

Figure 4.3: Google DriveFS folder in AppData and active DriveFS process in Task Manager 24

Figure 4.4: Key user activities (file sync, rename, deletion) executed inside the VM to generate DriveFS artifacts 26

Figure 4.5: Forensic imaging of the VM using FTK Imager, producing hash-verified E01 evidence for ingestion into Autopsy 27

Figure 4.6: Custom DriveFS parser placed in Autopsy's Python modules directory and detected during case setup 30

Figure 4.7: Execution of the Google DriveFS parser within Autopsy's ingest process, showing artifact tree generation 32

Figure 4.8: Parsed database evidence extracted from mirror_metadata.db and metadata_sqlite.db 33

Figure 4.9: Parsed logs and configuration files from structured_log_global, drive_fs.txt, and LocalPrefs.json 35

Figure 4.10: Cross-validation of plugin results against manual forensic queries in SQLite DB Browser 40

Figure 5.1: Conceptual diagram – bridging gap (Legacy → DriveFS → Autopsy Plugin) 47

Table of Tables

Table 2.1: Summary of key Google Drive client artifacts and forensic significance 9

Table 2.2: Legacy vs modern DriveFS artifacts 11

Table 2.3: Forensic tools comparison 14

Table 3.1: Testing and validation plan 18

Table 3.2: Project challenges and mitigation strategies 20

Table 4.1: Legacy vs modern artifact verification during testing 28

Table 4.2: Artifact mapping: source files, fields, and Autopsy attributes.. 31

Table 4.3: Plugin performance benchmark (manual vs automated) 35

Table 5.1: Research question evaluation 45

Table 5.2: Literature vs project contributions 48

Table 5.3: Strengths vs limitations of the developed plugin 49

Chapter 1: Introduction

Cloud storage is now integrated into modern cyber operations, with Google Drive alone surpassing two billion monthly users [1]. Although its usefulness is clear, forensic analysts face growing difficulties in identifying malicious activity on these platforms. Threat actors exploit Google Drive for data theft, malware staging, and insider theft because activities are harder to trace than on local systems [2]. Additionally, regulations like the General Data Protection Regulation (GDPR) require investigators to handle evidence in ways that respect privacy and proportionality, adding both technical and ethical challenges.

Forensic research has long highlighted the importance of local client-side artifacts. Nonetheless, [2] states most existing studies focus on legacy artifacts such as snapshot.db and sync_log.log, which are no longer present in recent versions of Google Drive for Desktop (DriveFS v40+). This creates a gap between research and practice: investigators may overlook vital evidence unless tools are updated to match platform changes. To address this, the dissertation introduces an Autopsy ingest module capable of automatically parsing modern DriveFS artifacts, including metadata_sqlite.db, structured_log_global, drive_fs.txt, and LocalPrefs.json.

By bridging this gap, the project contributes to both cybersecurity and digital forensics. It enables quicker threat detection, improves forensic readiness for insider incidents, and provides an open-source alternative to expensive commercial suites. Additionally, it demonstrates how open tools can remain compliant with ethical and legal standards, such as the GDPR and Google's Terms of Service.

A key challenge in cloud forensics is obtaining data directly from cloud providers. Legal, privacy, and jurisdictional barriers often prevent investigators from accessing server-side logs or content [3][4]. This often makes local artifact examination not just an alternative but a prerequisite to justify cloud data access requests, as stressed in Synctriage-based triage models [5]. Although cloud providers do keep activity logs and user data, these are rarely accessible without lengthy legal procedures (if at all). Consequently, forensic practitioners have shifted to client-side forensics analysing the traces left on user devices by cloud services. Every time a user syncs files through a desktop client or accesses cloud data from a computer or smartphone, residual artifacts are created on that device [2]. These artifacts may include local copies of files, configuration databases, application logs, cache files, registry entries, and more.

Previous research in cloud storage forensics consistently demonstrates that client-side artifacts provide extensive evidence about user actions, even when the main data is stored remotely. For example, investigators have retrieved information such as cloud file names, upload and download timestamps and records of deleted files by examining local databases and log files from services like Google Drive and Dropbox. At core, the user's device often holds the key to uncovering cloud-related activities when direct access to cloud data is not available.

Definitely, over the past decade, several studies have shown the forensic importance of these client-side remnants. Early work by [2] on Google Drive and others on Dropbox and OneDrive laid the foundation by identifying where cloud clients store their data on a PC and what information can be recovered [7]. These and later investigations revealed that even if files only "live" in the cloud, the act of syncing or interacting with them leaves behind logs, cache databases, and other traces on the local machine. Importantly, investigators need to know how to find and interpret these artifacts, because valuable evidence can be missed without a "contemporary understanding of the location and type of data remains left behind by cloud storage users on the devices they use" [7].

The Synctriage model proposed by [5] demonstrates how synchronisation logs from both desktop and mobile endpoints can be triaged to reconstruct cross-device timelines, aligning with this project's goal of automating such reconstructions.

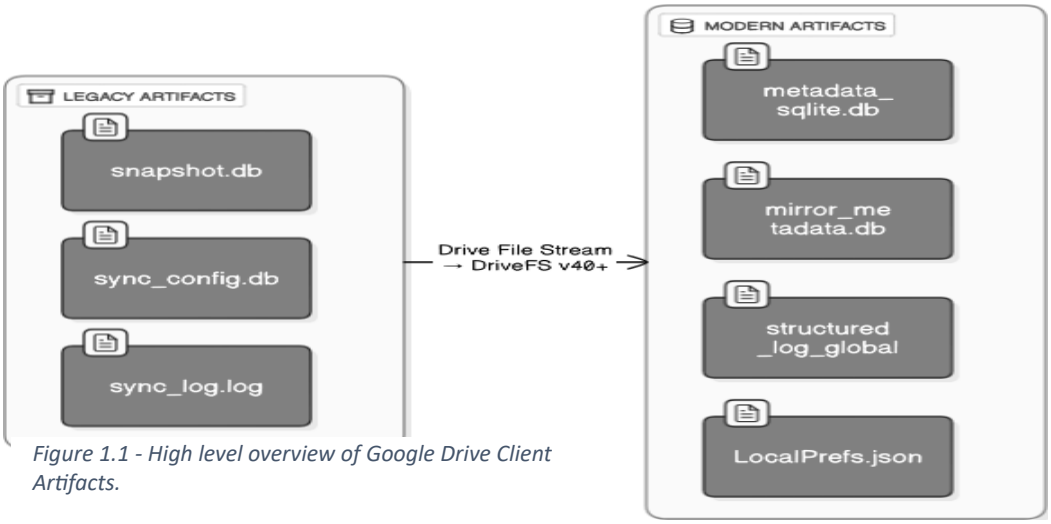


Figure 1.1 - High level overview of Google Drive Client Artifacts.

Despite these advances, most current studies assume legacy artifacts like snapshot.db or sync_log.log still exist. As shown in Chapter 2, these artifacts are absent in modern Google Drive for Desktop (DriveFS), replaced by databases such as metadata_sqlite.db and JSON-based logs. This

evolution highlights a gap between academic research and current forensic practice, which this project aims to bridge.

In practice, this means understanding that Google Drive's sync client maintains an SQLite-based metadata database (e.g., metadata_sqlite.db, mirror_metadata.db) and structured JSON or text-based logs (e.g., structured_log_global, drive.fs.txt). The forensic community's literature consistently agrees that such client-side evidence can allow for the reconstruction of events, such as file uploads, sharing and deletions, with detailed accuracy.

However, traditional artifacts like snapshot.db and sync_log.log, often cited in earlier research, have been phased out in modern Google Drive for Desktop (DriveFS). These are now replaced by new databases and structured logs, which means that old forensic tools may no longer cover everything. This project therefore focuses on identifying and parsing the new DriveFS artifacts to maintain forensic visibility despite updates to the client.

However, applying this knowledge in real casework is complex. Extracting these artifacts often requires manual analysis or ad-hoc scripting, and their locations or formats can change with software updates. This creates a significant gap in the current capabilities of forensic tools. While commercial suites are starting to include cloud artifact parsers, many popular open-source platforms (such as Autopsy) lack built-in support for cloud storage artifacts. For instance, an investigator using Autopsy on a seized hard drive would not automatically get a report of "Google Drive activity" unless they manually explore the file system to find the relevant files. This manual process is time-consuming and can be easily overlooked, particularly if the examiner is not already familiar with what to look for. This limitation has been frequently noted in surveys of forensic practitioners, with analysts pointing out that although commercial tools like Magnet AXIOM have begun adding parsers for DriveFS, open-source options such as Autopsy still lack support. This gap reduces accessibility, especially for investigators with limited budgets or in academic settings.

As [8] notes, if examiners do not know about the different types of cloud artifacts and where to find them, "they could miss critical information during an investigation" [7].

Given the stakes, since cloud storage may contain evidence of serious crimes or substantial data leaks, there's a clear need to simplify and organise the analysis of cloud client artifacts. Specifically, automating the detection and parsing of these artifacts would significantly enhance

forensic investigations by preventing the loss of cloud-related evidence and accelerating the analysis process.

1.1 Motivation and Aim

Based on the above context, this project aims to fill the tooling gap by creating an Autopsy module for Google Drive artifacts. Google Drive is chosen not only because of its popularity and forensic importance but also as a common example of cloud storage platforms often encountered by investigators. The goal is to develop an Autopsy plugin (ingest module) that can automatically detect Google Drive client artifacts on a Windows system, extract relevant evidence such as user account information, lists of synchronised files, and sync activity logs, and clearly display this data to the examiner. This way, the module will translate previous research into a practical tool. It will assist investigators in answering questions like: Was Google Drive used on this machine? What files were synced or accessed, and when? Were any files deleted from the cloud using this device?

Ultimately, the project's success will be judged by how effectively the developed module can reconstruct a user's Google Drive usage history from a forensic disk image, thus illustrating the importance of client-side cloud artifacts in a real investigative workflow.

In addition to its technical advances, this project also addresses broader professional needs by:

- Reducing examiners' dependence on manual SQLite or log analysis.
- Showing that open-source frameworks can be adapted to support contemporary cloud artifacts.
- Offering a replicable methodology for education, research, and practical digital investigations.

1.2 Research Questions

Q1. What important forensic artifacts are produced by the Google Drive for Desktop client on Windows, and what user activity evidence can they uncover?
Q2. How can these artifacts be automatically detected, extracted, and analysed within the Autopsy forensic platform?
Q3. How effective is the developed Autopsy ingest module at reconstructing user activity timelines, and what are its strengths and limitations in forensic workflows?

Chapter 2: Literature Review

2.1 Introduction

Cloud storage forensics is becoming increasingly important in digital investigations as more people use services such as Google Drive, Dropbox, and OneDrive [9][10]. However, a major challenge in this area is that investigators often struggle to access data directly from cloud providers. This challenge mainly stems from legal restrictions, privacy concerns, and jurisdictional issues that block access to server-side logs and content [4][6]. For example, Adesina et al. [4] note that while some users' activity is logged by providers, these records are not easily obtainable due to legal and privacy frameworks.

Due to these barriers researchers have gradually focused on client-side forensics. This involves examining evidence stored on the user devices that have interacted with the cloud [10][2]. This method works well because syncing apps and web browsers naturally leave behind traces like files, logs, registry entries and caches which can be analysed without needing help from the cloud provider [6][9]. These local artifacts often prove more accessible and legally viable for forensic examination.

As [11] note that while official APIs can be used to collect some cloud-side evidence they depend heavily on the cooperation of service providers and may not always yield complete or timely results. In contrast, [6] emphasizes that artifacts stored on local devices are often the most practical evidence source because server-side data is frequently inaccessible. That's why many studies [6][11] agree that an effective cloud investigation strategy should always include a thorough search of the suspect's devices for any sign of cloud storage activity, along with any available cloud-side data.

This is exactly the gap that this project aims to fill. The Google Drive Autopsy plugin developed for this dissertation will focus on automatically detecting and parsing artifacts left on the user's machine, which should speed up the process and reduce the risk of missing evidence. The literature also emphasizes the need for well-defined forensic procedures.

The literature emphasizes the importance of using structured forensic procedures when handling evidence. For instance,[12] point out the advantages of employing formal methodologies, such as the NIST Collection, Examination, Analysis, and Reporting model, to maintain the integrity of the evidentiary chain. This approach is particularly significant in cloud environments, where forensic artifacts may be dispersed across multiple files and directories on the device.

In summary, existing research indicates that when server-side data is unavailable or restricted, a careful client-side investigation becomes crucial [6][11]. This demonstrates the practical importance of creating a tool that automates this process within Autopsy, as proposed in this dissertation.

This chapter begins with a review of existing research on legacy Google Drive artifacts, then discusses their modern replacements in DriveFS. It summarises previous work on cloud client artifacts, reviews forensic tooling, and discusses open-source limitations. The chapter ends with a synthesis of the findings and highlights a research gap, which directly motivates the implementation detailed in Chapter 4."

2.2 Prior Work on Cloud Client artifacts

Several studies have investigated the digital traces left by cloud storage apps on local devices. Early research in this area was conducted by [11], who performed forensic analyses on services like Dropbox, Google Drive and Microsoft OneDrive (previously SkyDrive). Their results showed that even when files are stored in the cloud, syncing or accessing them from a device leaves various artifacts behind.

As [7] and [13] have identified critical areas within Windows-based environments where Google Drive and similar cloud clients often leave behind forensic traces. These traces include configuration databases, synchronisation records, and user credentials stored in registry keys. [13] emphasize the importance of maintaining forensic readiness by focusing on the structured retention of these artifacts, particularly within the AppData directory and relevant registry hives. These locations often preserve metadata such as timestamps, recent file interactions, and potential remnants of user authentication, reinforcing earlier findings reported by [11].

In these studies, a common pattern emerges: each cloud client leaves a distinct footprint on the user's system. Investigators typically use two methods to uncover these footprints. The first method involves targeted searches in known directories and registry paths (e.g., %APPDATA% and HKCU registry branches), while the second method involves broader disk and memory sweeps using forensic tools and keyword filters [11]. Here [13] supports this dual approach by recommending systematic monitoring of known Google Drive artifact locations and retention of deleted metadata.

As [3] and [7] recommend combining both methods: first acquiring known artifacts and then employing open-ended search techniques to minimize the risk of overlooking critical data.

[8] offers further insight by testing the resilience of cloud client artifacts under various scenarios. By simulating common user actions like uploading and deleting files, uninstalling the client, or running system cleaners, Chang demonstrates that even when users try to hide their cloud activities, many artifacts still remain on the system. This shows that client-side evidence can often be recovered with the right forensic tools and knowledge.

It is also important to distinguish between traces left by a dedicated cloud client and those left by just accessing services through a web browser. Using a browser to reach services like Google Drive or Dropbox may leave only limited traces, such as browsing history, cookies or downloaded files. In contrast, installing and using the official sync client generally produces a richer set of artifacts, including local databases (SQLite) and detailed sync metadata [11][12]. Most researchers agree that both access methods should be examined in an investigation; however, the agreement is that client software artifacts offer more detailed timelines of file operations, which are especially useful for reconstructing user activity.

An important advancement in cloud forensics is the Synctriage approach proposed by [5], which focuses on the forensic triage of cloud-based data exchanged between mobile and desktop environments. Their study demonstrates how artifacts from both endpoints, including remnants from mobile devices and desktop sync logs, can be utilised to reconstruct shared cloud interactions. While their work highlights the importance of collecting cross-device evidence, it also reinforces the broader principle that client-side artifacts hold substantial forensic value in cloud investigations. This concept of triage aligns with the project's objective to automate the detection and correlation of local Google Drive artifacts within Autopsy, thereby enabling the swift reconstruction of user activity timelines without requiring server-side access.

This project builds on these findings by focusing on the local artifacts produced by the Google Drive desktop client. By analysing the footprints documented in previous research, this Autopsy plugin will identify the most important evidence sources and automate what would otherwise be a lengthy manual process.

2.3 Google Drive Artifact Analysis in Existing Research

Google Drive has been widely studied in forensic research, consistently showing that it leaves significant evidence on computers using its sync client. One of the earliest detailed studies was by [2], who examined Google Drive on Windows PCs and iPhones. They demonstrated that installing and using the Google Drive client creates evidence in various forms, including the local file system (a "Google Drive" folder with synced files), Windows registry entries, link files to recently accessed cloud files, prefetch files and logs [14]. For example, they found that the client's prefetch file (googledrivesync.exe-.pf*) could reveal when the application was last run, and registry keys confirmed that Google Drive was installed and configured to run at startup [2][4]. These early findings established a baseline for confirming the presence of Google Drive on a device and reconstructing user activity.

Recent research has explored deeper into important client-side artifacts. [7] and [12] highlight that database files like sync_config.db and snapshot.db are valuable sources of forensic data. The sync_config.db file contains the user's email address and the path to the local sync folder, while snapshot.db provides detailed metadata about all synced files. This metadata includes file names, unique identifiers, sizes, checksums and timestamps.

As [14] demonstrated that deleted entries can sometimes be recovered from the database's write-ahead log (WAL) or disk slack space, making snapshot.db a valuable source even when files have been removed from the machine.

Another important artifact is the sync_log.log file which records the Google Drive client's actions in order. As [12] describe this log as containing events of creation, deletion and modification. By analysing sync_log.log, investigators can create detailed timelines of cloud-related activities, cross referencing it with snapshot.db to determine exactly when files were created, modified or deleted and whether the actions occurred locally or through the web interface [14]. This approach helps differentiate between local deletions and those initiated online.

Recent research by [13] emphasizes the importance of preparing forensic environments to identify and recover structured remnants created by cloud storage clients like Google Drive. Their study focuses on analysing post-activity artifacts on Windows 10 systems, demonstrating how databases such as snapshot.db, sync_config.db, and various registry entries can be

used to reconstruct user interactions, even after the Google Drive client has been uninstalled or files have been deleted.

They stress the need to prioritise persistent, structured artifacts, such as configuration files and synchronisation logs, over volatile system metadata. In particular, [13] highlight that crucial forensic evidence often resides in the AppData directory and registry hives, which retain timestamped metadata, file paths, and user credentials. These findings reinforce the consensus that effective cloud forensic reconstruction relies on reliable client-side data sources, underscoring the need for tools that can automatically parse such artifacts.

To understand these details, Table 2.1 summarises the main Google Drive client artifacts identified in various studies and explains their forensic significance.

Table 2.1 - Summary of key Google Drive client artifacts and forensic significance

Artifact	**Forensic Significance**
Local Sync Folder	Stores synced files. Shows which files were uploaded to the cloud. Timestamps may differ from cloud times.
sync_config.db	Contains the user account email and the local sync directory path. Confirms which account was associated with the machine.
Snapshot.db	Lists all files recognised by the Google Drive client with metadata (file names, IDs, timestamps and MD5). Displays files that are currently present or were deleted.
sync_log.log	Plain text log of file actions (uploads, deletions and modifications) with timestamps. Useful for reconstructing timelines.
Additional Databases	Newer versions (Drive File Stream/Drive for Desktop) include *cloud_graph.db*, *global.db*, and *global_preferences.db* for folder structures and settings.
Windows Registry Keys	Verify installation, configuration and startup behaviour. It may

	show when the client was installed and used.

After reviewing these sources, researchers agree that combining snapshot.db and sync_log.log is especially effective for reconstructing detailed user actions. Metadata within these databases is often more reliable than Windows file system timestamps. For example, if a file is deleted and later restored from the cloud, the Windows file system might show a new creation date, but snapshot.db retains the original creation timestamp [12].

Finally, studies by [7] and [8] demonstrate the resilience of these artifacts. Even when users uninstall the Drive client or run system cleaners like CCleaner, significant traces often remain in the local sync folder, databases and registry keys. This persistence highlights the need for tools that can reliably locate and interpret Google Drive artifacts. It directly supports the project's aim: to build an Autopsy module that automates the recovery and correlation of these artifacts and then provides investigators with clear evidence of cloud storage usage.

2.4 Modernisation of Google Drive Client Artifacts in DriveFS

In earlier forensic literature, the main focus was on legacy Google Drive artifacts such as `snapshot.db`, `sync_config.db`, and `sync_log.log`. These SQLite databases and log files, usually found under `%APPDATA%\Local\Google\Drive\user_default\`, provided insights into file activities, synchronisation status, and configuration data [6], [11], [12].

Although artifacts like [snapshot.db, sync_config.db, sync_log.log] offered detailed forensic insights in the Google Drive File Stream era, these files are no longer generated by Drive for Desktop. Relying on older techniques may lead to missing crucial evidence, as user activity now produces data in different files such as metadata_sqlite.db and structured_log_global. This shift requires updated analysis approaches, as validated during the project's implementation phase.

With the move to **Google Drive for Desktop (DriveFS v40+)**, there has been a major change in how client-side artifacts are located, structured, and named. Modern DriveFS artifacts, usually located under `%LOCALAPPDATA%\Google\DriveFS\`, now include:

- `metadata_sqlite.db` and `mirror_metadata_sqlite.db` are successors to `snapshot.db`. They track file states, versions, paths, Drive

identifiers, and include information on deleted or tombstoned items files.

- structured_log_global, a unified JSON log that records sync operations, user actions, and file events with timestamps. This effectively replaces sync_log.log.
- The old sync_config.db, which stored account credentials, folder mappings, and configuration data, no longer exists in DriveFS. Its functions are now spread across several modern files, such as LocalPrefs.json, which keeps track of account linkage and sync preferences, startup_trace.json, which records runtime session settings, and Windows Registry keys under NTUSER.DAT, which manage Drive mappings and installation paths. Together, these artifacts provide similar coverage to what sync_config.db used to hold, though in a more fragmented way.
- drive_fs.txt and parent.txt, plain text logs that map relationships between items and display low-level sync details.

During this project's implementation (see Chapter 4), these updated artifacts were directly observed in controlled virtual machine experiments. For example, file operations such as renaming, deletion, and restoration appeared not in snapshot.db or sync_log.log but in mirror_metadata_sqlite.db and structured_log_global, confirming the obsolescence of the legacy artifacts. Similarly, LocalPrefs.json was found to contain account and sync configuration data, emphasising its modern relevance as a partial replacement for sync_config.db.

This change has important effects. First, it shows that forensic tools based on old artifacts may not find user activity accurately anymore. Second, it highlights the need to create updated modules, like the Autopsy ingest plugin built in this project, to automatically analyse and display DriveFS's new artifact set. This allows investigators to keep building detailed timelines of Google Drive activity despite these structural changes.

Table 2.2 - Legacy vs modern DriveFS artifacts

Legacy Artifact	**Function**	**Modern Replacements in DriveFS**	**Forensic Notes**
snapshot.db	File listings, sync status, metadata	metadata_sqlite.db, mirror_metadata_sqlite.db	Offers detailed file records, timestamps, deletion flags,

			and folder structure.
sync_log.log	Synchronisation logs, operations, and errors.	structured_log_global, drive_fs.txt, parent.txt	JSON-based logs with detailed operations (upload, remove, rename), account linkage, and error events.
sync_config.db	Stored account linkage, sync folder mappings, and configuration	LocalPrefs.json, startup_trace.json, Windows Registry (NTUSER.DAT)	Distributes configuration: LocalPrefs.json stores account and folder details; startup_trace.json records runtime session settings; Registry keeps install path and mappings.
—	—	thumbnails/ and content_cache/	Stores cached files, previews, and chunks for recovery.
—	—	metrics_store_sqlite.db	Tracks internal performance and operational metrics.

Note: Unlike snapshot.db and sync_log.log, which have clear modern equivalents, the function of sync_config.db has been dispersed across multiple artifacts. This scattered configuration increases forensic workload, as investigators now need to link JSON files, startup traces, and registry hives to reconstruct account and sync settings. These findings also guided the validation approach outlined in Chapter 4, where both automated

parsing and manual cross-verification were used to verify the forensic significance of these artifacts.

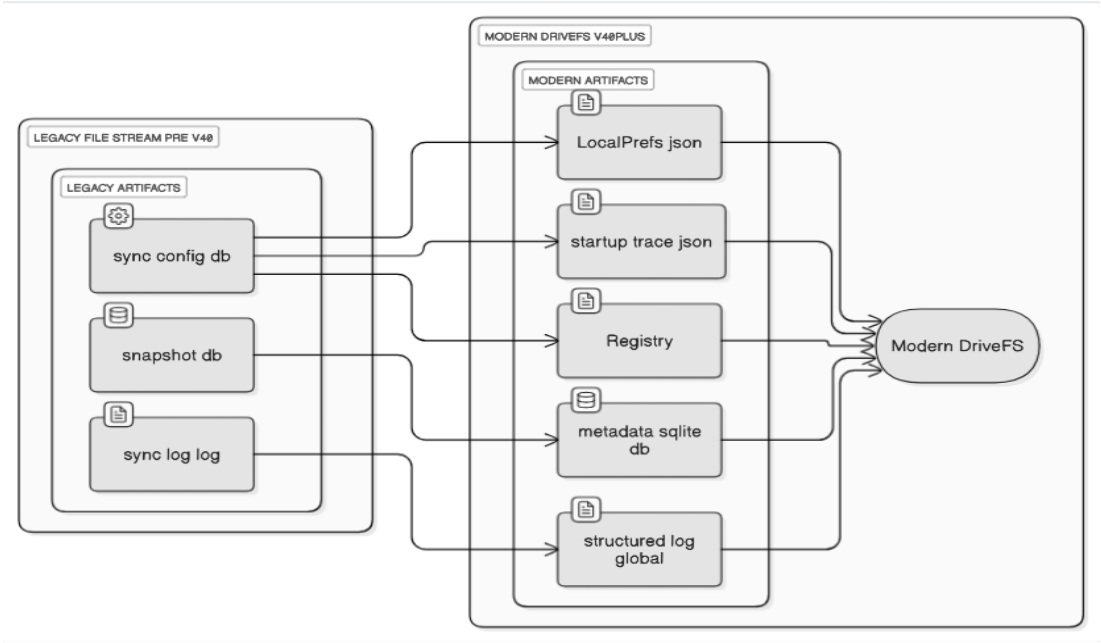


Figure 2.1 - Detailed Evolution of Google Drive client artifacts from legacy File Stream to modern DriveFS.

2.5 Tools for Artifact Analysis and Current Gaps

Over the years, forensic researchers and practitioners have explored various methods to extract and analyse artifacts left by cloud storage services. Early investigations commonly employed manual approaches where analysts wrote custom scripts or inspected files, such as SQLite databases, using generic tools like DB Browser for SQLite [2][7]. Although effective, this method requires considerable effort and expertise, particularly when dealing with multiple data sources like snapshot.db, sync_log.log or cloud_graph.db.

To alleviate this workload, commercial forensic tools like Magnet AXIOM and Internet Evidence Finder (IEF) have incorporated modules designed to identify and extract cloud-related data, including artifacts from services like Dropbox and Google Drive [12]. These platforms offer user-friendly interfaces and fit well within investigative workflows. However, their proprietary nature and licensing costs can render them inaccessible to smaller labs, students or open-source environments.

In the open-source domain, tools like ArtiFast and Belkasoft Evidence Center provide partial support for parsing Windows-based artifact data,

with some modules also including cloud traces [15]. These tools reflect the community's growing recognition of the necessity to automate the extraction of cloud artifacts. However, open-source frameworks continue to exhibit significant gaps, especially when dealing with the evolving artifact formats from newer versions of cloud clients.

One widely used open-source tool, Autopsy provides strong support for general disk forensics but currently lacks built-in functionality to automatically parse Google Drive artifacts. Investigators must manually locate files such as snapshot.db or sync_log.log on a suspect's device and analyse them using external tools, which introduces delays and increases the risk of human error. While a few community-built Autopsy modules exist on platforms like GitHub, many are unmaintained or only support older versions of Google Drive [16].

Moreover, cloud providers often update their desktop applications, altering where and how artifacts are stored. For instance, Google's transition from the classic Drive client to Drive File Stream and Drive for Desktop introduced new database files like cloud_graph.db, global.db, and global_preferences.db, which are often stored in different file paths or structured differently [7][12]. Forensic tools that are not regularly updated may fail to identify these new sources of evidence, potentially leading to incomplete investigations.

These challenges underscore a persistent gap in current tools: the absence of a reliable, open-source solution to automate the parsing of Google Drive artifacts in forensic environments. This dissertation aims to address that gap by developing an Autopsy ingest module capable of automatically detecting, parsing, and presenting Google Drive artifacts from both legacy and current clients in a forensically sound manner. This contribution supports ongoing efforts in digital forensics to streamline workflows, enhance artifact visibility, and provide accessible tools for investigators, regardless of their budget or licensing constraints.

Table 2.3 - Forensic tools comparison

Tool	**Type**	**Cloud Artifact Support**	**Google DriveFS Coverage**	**Limitations**
Magnet AXIOM	Commercial	Yes	Partial (legacy and some DriveFS)	Expensive and closed-source

Belkasoft EC	Commercial	Yes	Limited DriveFS parsing	Costly
ArtiFast	Open-Source	Partial	Outdated (no DriveFS)	Poor maintenance
Autopsy	Open-Source	No native support	Requires manual DB/log parsing	Misses DriveFS artifacts

Recent contributions by [13] and [12] illustrate that DriveFS continues to evolve rapidly, with new artifacts such as metadata_update.db and metrics_store.db gaining forensic significance. However, most studies still rely on manual SQL queries or narrow validation frameworks, which limits scalability to enterprise contexts. Few incorporate quantitative metrics or automated benchmarking, leaving a gap that this project explicitly addressed in its evaluation strategy.

2.6 Summary of findings and research gap

Existing literature clearly establishes that cloud storage platforms, including Google Drive, leave behind a variety of artifacts on client devices. These artifacts, such as snapshot.db, sync_log.log and various registry keys, provide valuable insights into user activity. Studies have shown that these remnants often persist even after attempts to delete them or remove the client application [7][12]. Researchers have developed systematic methods for discovering these traces, typically combining location-specific file system searches with broader disk and memory analysis techniques [7][11].

While several tools now support cloud artifact analysis, limitations persist, particularly in the open-source ecosystem. Commercial forensic tools like Magnet AXIOM and Belkasoft can parse Google Drive artifacts to some extent, but access to these platforms is limited by cost and licensing constraints [15].

Conversely, widely used open-source tools such as Autopsy offer excellent support for general digital evidence analysis but still lack native support for parsing Google Drive client artifacts [16]. Although some community-created Autopsy modules exist, they are often incomplete, outdated or incompatible with modern Google Drive clients like Drive for Desktop, which introduced new databases such as cloud_graph.db and global.db [7][15].

This gap in functionality presents a significant opportunity. There is a clear need for a dedicated, actively maintained Autopsy module that automates the identification and parsing of forensic artifacts associated with Google Drive's sync clients. Such a tool would reduce reliance on manual analysis, streamline investigation workflows, and enhance the reliability and reproducibility of forensic findings. By building on established forensic principles, such as structured artifact extraction [12], forensic triage [5] and timeline reconstruction, this project seeks to address these limitations directly.

The proposed plugin will focus on parsing both legacy and current Google Drive artifacts, helping investigators reconstruct detailed user activity without needing server-side access. In doing so, it aims to contribute a practical and technically sound tool to the broader field of cloud storage forensics, promoting accessibility, automation and forensic integrity.

Accordingly, this contributes an Autopsy ingest module focused on DriveFS artifacts. By automating the detection and parsing of metadata databases, structured logs, and configuration remnants, it directly tackles the limitations found in previous research and current forensic tools.

A further limitation in prior research is the methodological rigor of evaluation. Most existing studies lack explicit reporting of accuracy, precision, or reproducibility when validating forensic tools. By incorporating error-handling tests and cross-validation, this dissertation contributes a more structured framework, directly addressing these shortcomings.

Chapter 3: Project Plan

3.1 Introduction

To effectively address the problem identified in earlier chapters, this project adopts the Design Science Research (DSR) methodology. DSR is particularly suitable for developing innovative digital forensic tools, as it focuses on the structured creation and assessment of functional artifacts that solve real-world problems [17]. This structured approach includes six main phases: (1) problem identification and motivation, (2) defining solution objectives, (3) design and development, (4) demonstration, (5) evaluation, and (6) communication, as proposed by Peffers et al. [18].

This project integrates these steps into an iterative development cycle, enabling early testing and refinement of the Autopsy module through agile principles. The main reason for using DSR is its ability to connect practical implementation with scholarly contribution by creating a validated forensic tool for parsing Google Drive artifacts.

3.2 Project stages and Activities

The project is divided into clearly defined stages that build on insights from the literature review and problem definition.:

- **Requirement Analysis and Planning:** In this initial phase, findings from Chapters 1 and 2 are synthesised to identify the core features and requirements of the plugin. Key artifacts of interest include both legacy and modern Google Drive client traces. Legacy artifacts, such as snapshot.db, sync_config.db, and sync_log.log, offer historical context, while modern DriveFS artifacts, like metadata_sqlite.db, mirror_metadata_sqlite.db, structured_log_global, and LocalPrefs.json, are the primary focus of this study. Collectively, these artifacts allow reconstruction of file activity, sync status, account linkage, and user actions within the DriveFS environment. Ethical considerations are also covered at this stage through completion of the University of York ethics checklist.
- **Data Collection and Environment Setup:** Controlled test environments will be configured on virtual machines (Windows/macOS), where the Google Drive client will be installed and file operations will be performed. These actions will generate the forensic artifacts needed for plugin development. Imaging tools like FTK Imager will be used to capture data in a forensically sound manner [12].

- **Artifact Analysis and Reverse Engineering:** The contents of the artifacts will be analysed using the SQLite DB Browser and validated through test operations. The schema of each database will be reverse-engineered to understand the relevance of its fields. Prior research [12], [6], [5] and exploratory techniques such as SQL querying, hex-level browsing and snapshot comparison will be employed to classify the semantics of the fields. This phase also separates deprecated artifacts (such as snapshot.db, sync_log.log, sync_config.db) from their modern DriveFS replacements (metadata_sqlite.db, mirror_metadata_sqlite.db, structured_log_global, LocalPrefs.json, startup_trace.json), ensuring that the Autopsy plugin targets current Google Drive client structures instead of outdated ones. This guarantees that RQ1 is fulfilled by methodically pinpointing artifacts that demonstrate file operations, sync status, and user settings.
- **Plugin Design and Implementation:** The ingest module will be developed using Python 3.x, utilising Autopsy's Python API and blackboard services to extract and display data within the GUI [16]. The plugin will automatically extract, decode and display forensic evidence such as file timelines and sync actions within the Autopsy GUI. Git will be used to manage iterative development.

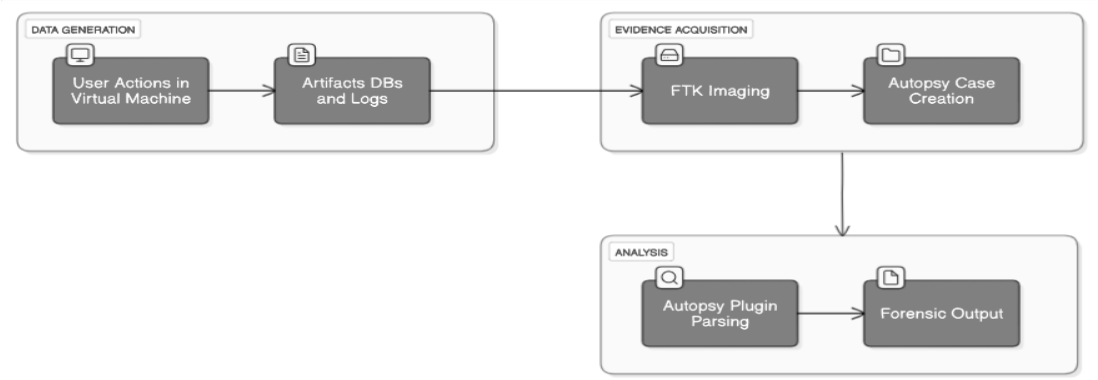


Figure 3.1 - Project stages (DSR framework).

- **Testing and Validation**: unit tests will be developed with pytest and functional testing will be performed on the dummy datasets to ensure correctness, coverage, and resilience. Autopsy's internal logs and feedback will support in iterative refinement.

Table 3.1 - Testing and validation plan

Test Type	Purpose	Researched Questions Addressed

Unit Tests	Ensure the parser extracts the correct fields	RQ2
Functional Tests	Validate plugin in the Autopsy ingest pipeline	RQ2, RQ3
Cross-Validation	Compare plugin results with manual DB/log inspection	RQ3
Performance Benchmark	Compare plugin versus manual analysis speed	RQ3

- **Evaluation:** The developed module will be benchmarked against manual forensic analysis to assess its time efficiency and completeness. If possible, gather feedback from peers or supervisors using established criteria from previous DFIR evaluations [15].
- **Documentation and Reporting**: The development process will be supported by comprehensive documentation, including code comments, a user README and the final dissertation write-up. The README will outline supported Drive versions and known limitations. Documentation will facilitate reproducibility and usability for other investigators.

3.3 Tools and Environment

Key technologies and essential tools include:

1. **Autopsy 4.x** – Forensic analysis platform with modular ingest modules [16].
2. **Python 3.x** – Primary language for module development.
3. **SQLite DB Browser** – Artifact inspection and schema analysis [12].
4. **Virtual Machines** – Isolated Windows for data generation.
5. **Git** – Version control for iterative development.
6. **FTK Imager -** Forensic imaging of virtual machine environments.
7. **Dummy Data Sets** – Simulated Google Drive activity for controlled analysis.
8. **Supporting libraries**- Python sqlite3, datetime, os, logging, etc

3.4 Research Question Alignment

Each project activity directly corresponds to the research questions outlined earlier:

RQ1: Explored during data collection and reverse engineering stages.
RQ2: Addressed through requirement analysis and identification of existing tool limitations.
RQ3: Assessed during plugin testing and final performance evaluation.

3.5 Challenges and Backup Strategies

Potential challenges and mitigations:

Table 3.2 - Project challenges and mitigation strategies

Challenges	Impact	Mitigation
Schema Changes	Parser may fail on new DriveFS versions	Lock version and modular queries
Artifact decoding	Base64/compressed fields may cause errors	Manual inspection plus decoding routines
Autopsy API learning	Delays development	Sandbox testing and restricted scope
Tool Compatibility Issues	Risk of instability	Stable versions of Autopsy (4.19)
Time limitations	Reduced scope	Prioritise core artifacts (metadata_sqlite.db, logs)

3.6 Ethical Considerations

This study adheres to the ethical standards of the University of York and will undergo review via the Computer Science Ethics System (PSEC). It does not involve personal or sensitive data; only dummy data created on isolated, researcher-controlled machines is analysed. To align with Google's Terms of Service, the research focuses solely on client-side DriveFS artifacts such as metadata_sqlite.db, mirror_metadata_sqlite.db, structured_log_global, LocalPrefs.json, and startup_trace.json. This approach respects user data ownership, avoids server-side access, and complies with the UK's "fair dealing" exception for academic purposes, as the researcher's machine is under full control and no server resources are accessed.

Design Science Research (DSR) was chosen as the main methodology because it aligns well with creating and assessing artifacts. Although other alternatives like CRISP-DM were evaluated, their focus on iterative data-mining cycles was less relevant. DSR offered a more straightforward structure for developing, testing, and validating the forensic module, while also meeting ethical and legal standards. This approach directly supports the research questions outlined in Chapter 1.

Chapter 4: Implementation and Results

4.1 Introduction

This chapter covers the implementation and evaluation of the proposed Autopsy ingest module for parsing Google Drive for Desktop (DriveFS) artifacts. Building on the phased methodology described in Chapter 3, the work moved from controlled artifact creation (RQ1) to plugin development and automation (RQ2), and then to validation and assessment (RQ3).

The project specifically targets the modern DriveFS artifact landscape, replacing legacy databases (snapshot.db, sync_log.log, sync_config.db) with current equivalents such as metadata_sqlite.db, mirror_metadata_sqlite.db, structured_log_global, and LocalPrefs.json. A controlled Windows 10 virtual environment was used to generate realistic test data, while FTK Imager and Autopsy provided forensic acquisition and analysis.

This chapter demonstrates how DriveFS artifacts can be systematically acquired, parsed, and presented to investigators using a combination of manual reverse-engineering and automated plugin testing within an open-source forensic workflow.

4.2 Development Environment and Toolchain

To ensure reproducibility and forensics validity, a dedicated virtual environment was established.

System Configuration:

- Host: Vmware Workstation
- Guest OS: Windows 10 Pro x64
- RAM: 4 GB | Disk: 60 GB | CPU: 4 cores
- Network: NAT (isolated from host)

Software Installed:

- Google Drive for Desktop (v45+)
- FTK imager (disk imaging, hash verification)
- SQLite DB Browser (manual DB inspection)
- Autopsy 4.19 with Python scripting support
- Python 3.x with VS Code (plugin development)
- 7-Zip, Notepad++ (Auxiliary tools)

A structured folder hierarchy (C:\DriveTesting) separated synced files, extracted artifacts, logs, and plugin outputs. User activities such as file

creation, deletion, and renaming were manually logged to establish a ground truth timeline for validation.

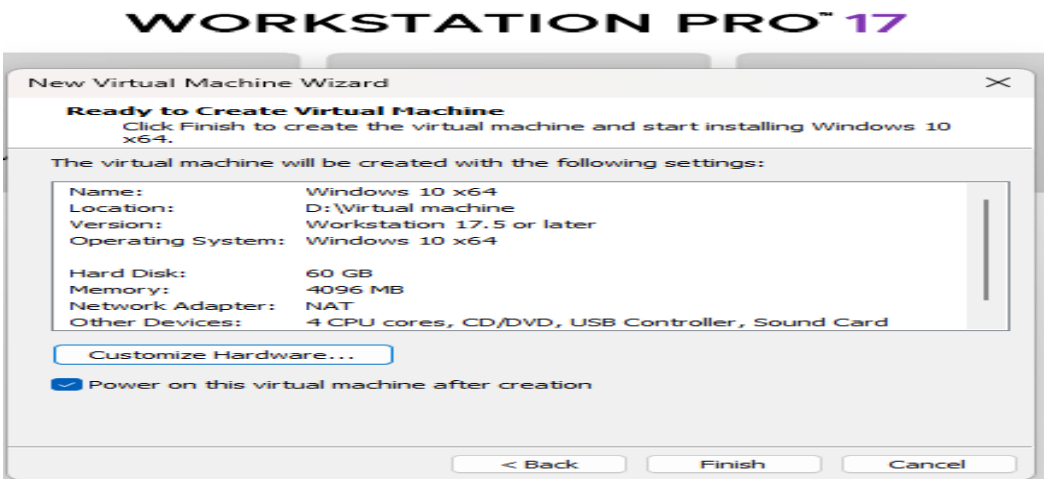


Figure 4.1 - Virtual machine creation in VMware Workstation.

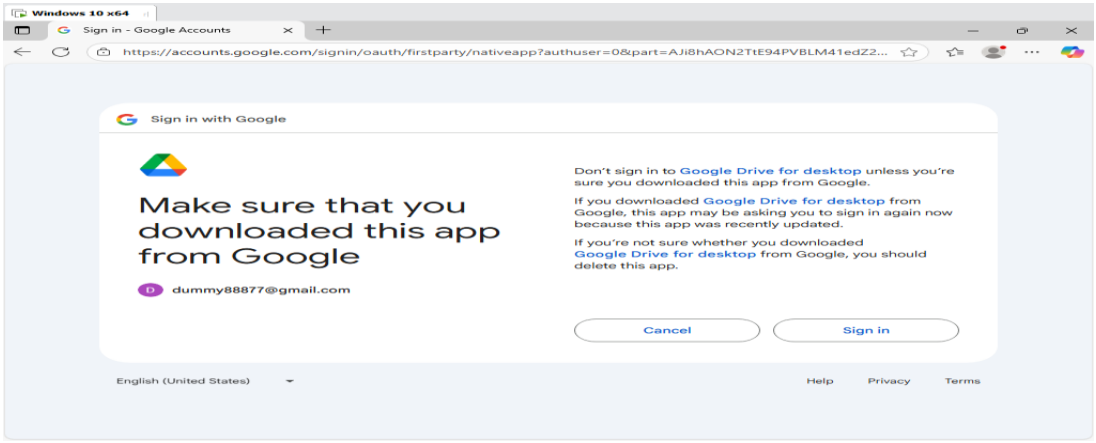
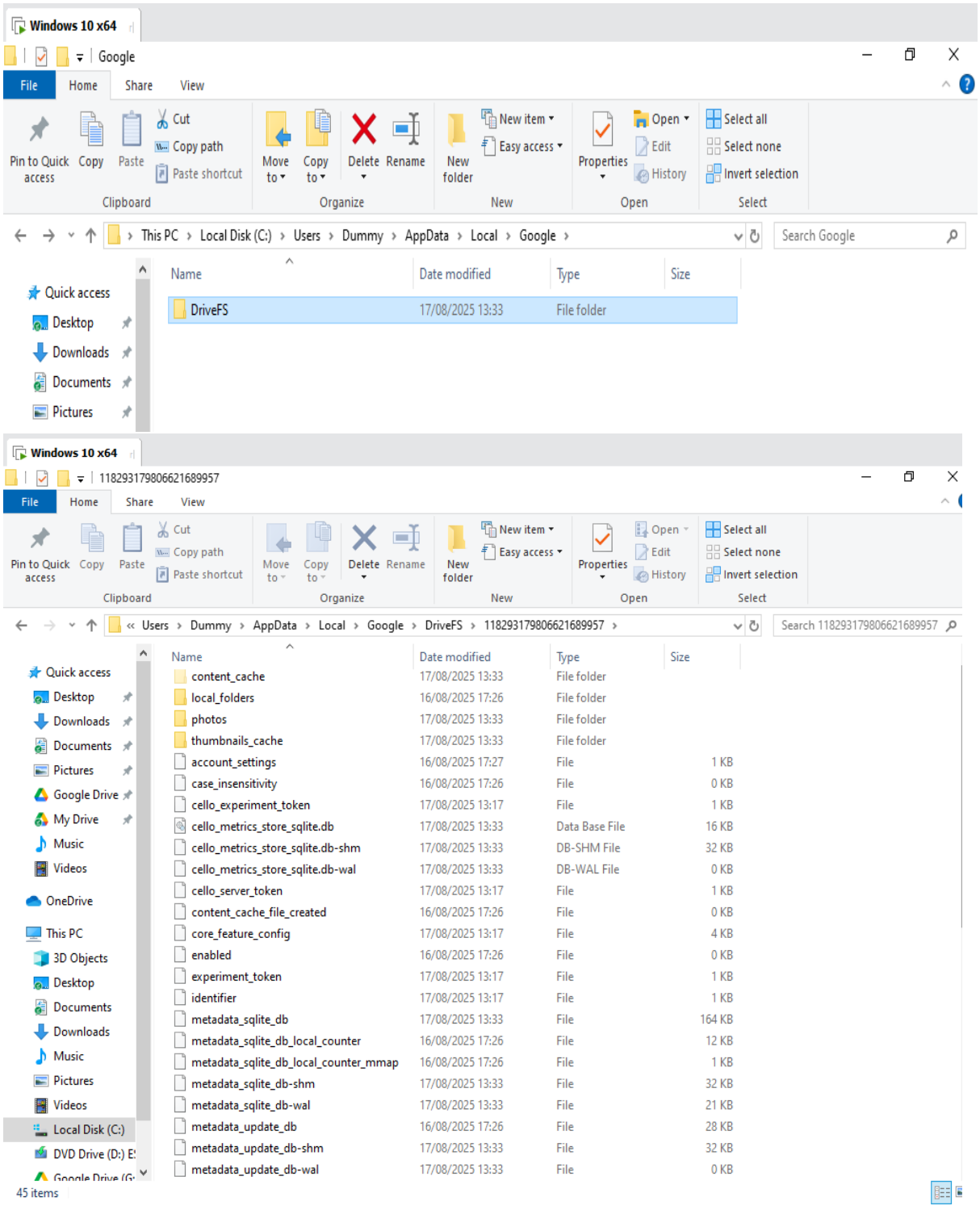


Figure 4.2 - Login of test Google Drive account using dummy credentials. Additional screenshots

Test Data Organisation: The DriveFS application process was confirmed to be running through Task Manager during testing. Evidence folders were located at %LOCALAPPDATA%\Google\DriveFS and contained key artifacts such as snapshot.db, structured_log_global, and cache files. These were later verified through manual inspection and plugin extraction.

Plugin Setup Context: To support later development, Autopsy's Python module directory was prepared for custom scripts, and the sqlite-jdbc.jar library was placed into Autopsy's modules folder to enable SQLite parsing. This ensured the environment was ready for ingest module integration before implementation began.


Windows 10 x64
Google
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Move to
Copy to
Delete
Rename
New folder
New item
Easy access
Properties
Open
Edit
History
Select all
Select none
Invert selection
Clipboard
Organize
New
Open
Select
This PC › Local Disk (C:) › Users › Dummy › AppData › Local › Google ›
Search Google
Quick access
Desktop
Downloads
Documents
Pictures
Name
Date modified
Type
Size
DriveFS
17/08/2025 13:33
File folder
Windows 10 x64
11829317980662168995 7
« Users › Dummy › AppData › Local › Google › DriveFS › 11829317980662168995 7 ›
Search 11829317980662168995 7
Google Drive
My Drive
Music
Videos
OneDrive
This PC
3D Objects
Local Disk (C:)
DVD Drive (D:) E
content_cache 17/08/2025 13:33 File folder
local_folders 16/08/2025 17:26 File folder
photos 17/08/2025 13:33 File folder
thumbnails_cache 17/08/2025 13:33 File folder
account_settings 16/08/2025 17:27 File 1 KB
case_insensitivity 16/08/2025 17:26 File 0 KB
cello_experiment_token 17/08/2025 13:17 File 1 KB
cello_metrics_store_sqlite.db 17/08/2025 13:33 Data Base File 16 KB
cello_metrics_store_sqlite.db-shm 17/08/2025 13:33 DB-SHM File 32 KB
cello_metrics_store_sqlite.db-wal 17/08/2025 13:33 DB-WAL File 0 KB
cello_server_token 17/08/2025 13:17 File 1 KB
content_cache_file_created 16/08/2025 17:26 File 0 KB
core_feature_config 17/08/2025 13:17 File 4 KB
enabled 16/08/2025 17:26 File 0 KB
experiment_token 17/08/2025 13:17 File 1 KB
identifier 17/08/2025 13:17 File 1 KB
metadata_sqlite_db 17/08/2025 13:33 File 164 KB
metadata_sqlite_db_local_counter 16/08/2025 17:26 File 12 KB
metadata_sqlite_db_local_counter_mmap 16/08/2025 17:26 File 1 KB
metadata_sqlite_db-shm 17/08/2025 13:33 File 32 KB
metadata_sqlite_db-wal 17/08/2025 13:33 File 21 KB
metadata_update_db 16/08/2025 17:26 File 28 KB
metadata_update_db-shm 17/08/2025 13:33 File 32 KB
metadata_update_db-wal 17/08/2025 13:33 File 0 KB
45 items

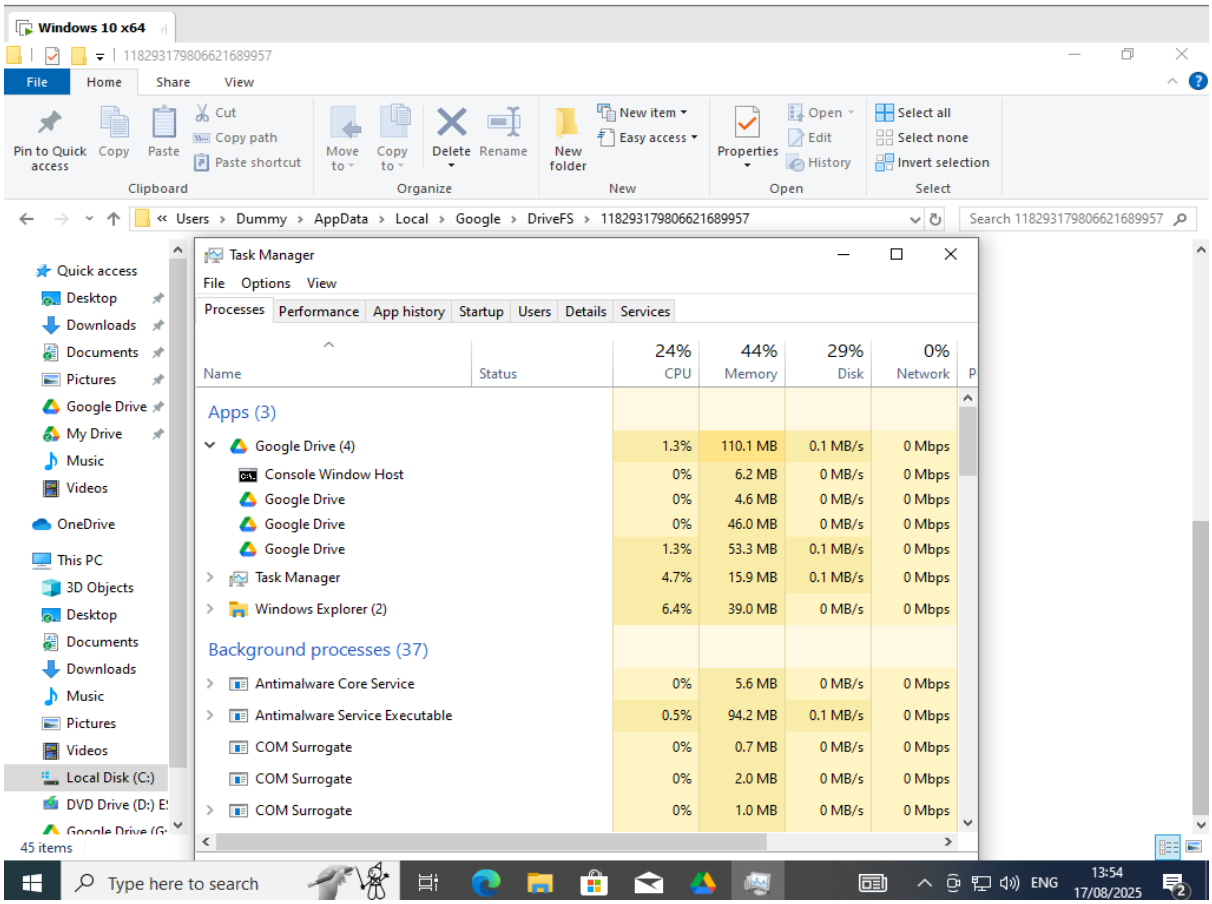


Figure 4.3 - Google DriveFS folder in AppData and active DriveFS process in Task Manager.

- Additional screenshots showing VM setup, DriveFS directory structures, and log folders are provided in **Appendix A (Figures A.1-A.6)** for reference.

4.3 Artifact Generation and Evidence Collection

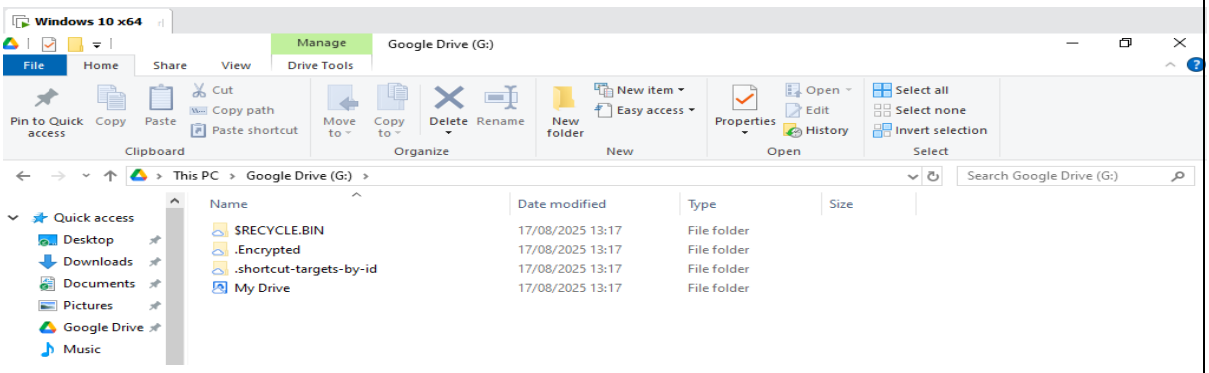
To mimic authentic DriveFS user behaviour, we ran a comprehensive suite of scenarios within the VM. These involved installing the Drive client, syncing files, editing content, deleting items (both locally and in the cloud), renaming folders, and uninstalling the client.

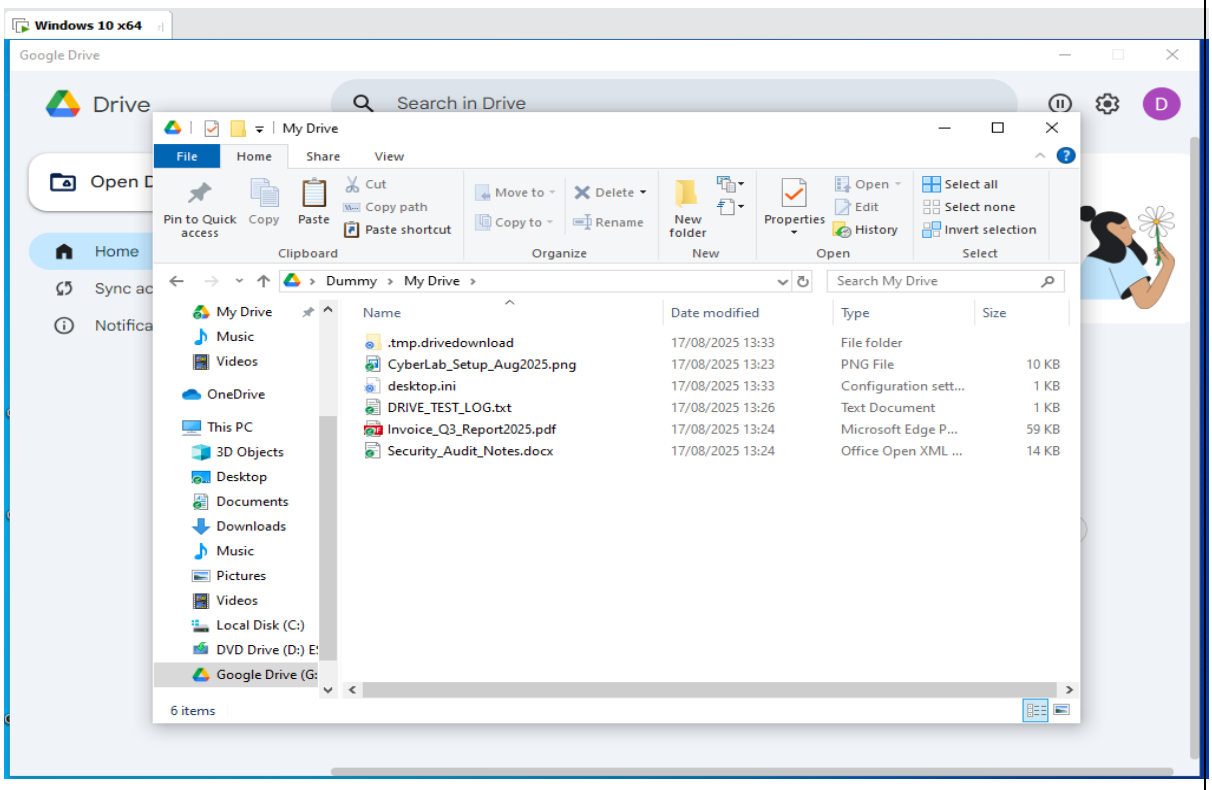
- S1: Initial Setup and login
- S2: File uploads and sync confirmations
- S3: File edits and version changes
- S4: Deletion (local and cloud-based)
- S5: Rename and move operations
- S6: Application termination and restart
- S7: Client uninstallation and post-analysis

Key artifacts extracted after each activity:

- SQLite databases: metadata_sqlite.db, mirror_metadata.db
- Logs: Structured_log_global, drive_fs.txt, parent.txt
- JSON configs: LocalPrefs.json, startup_trace.json
- Cached content: humbnails, stable parents, tombstoned items

Disk imaging was carried out using FTK Imager. Hash-verified. E01 images were subsequently imported into Autopsy.





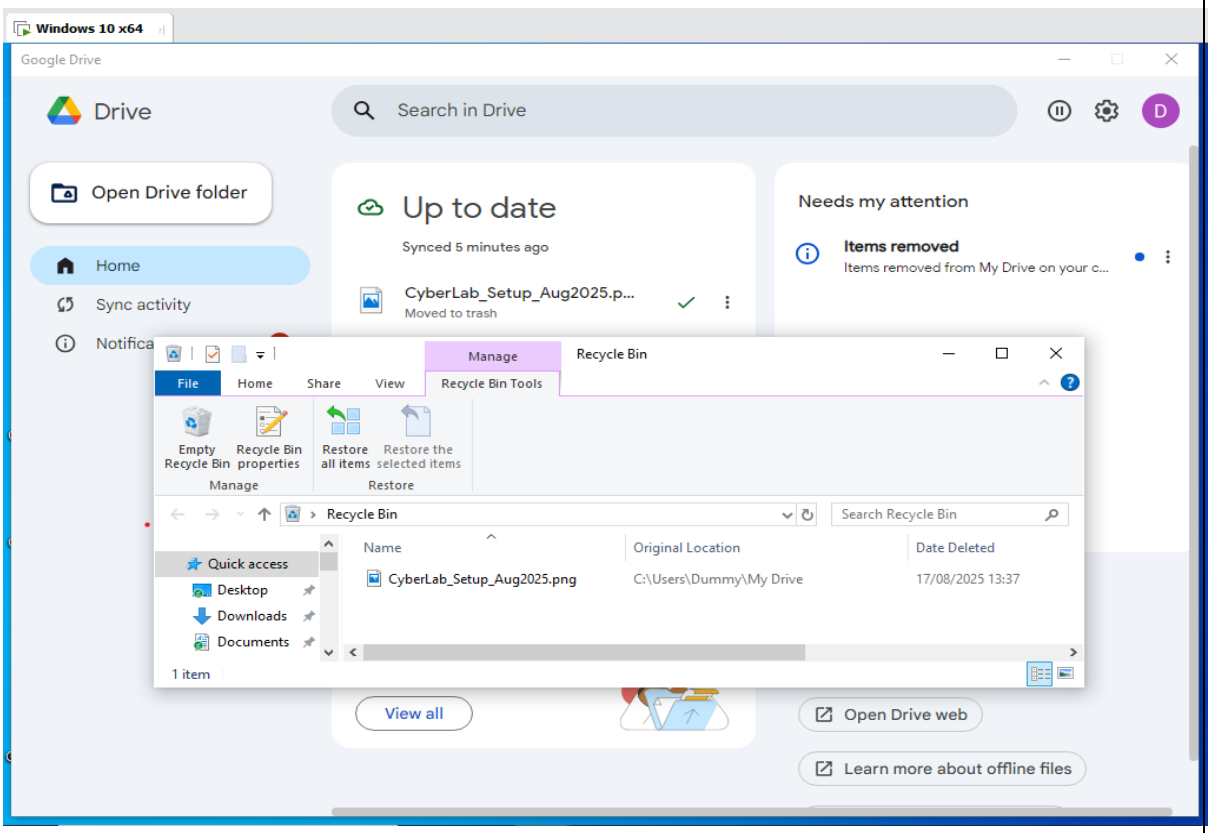
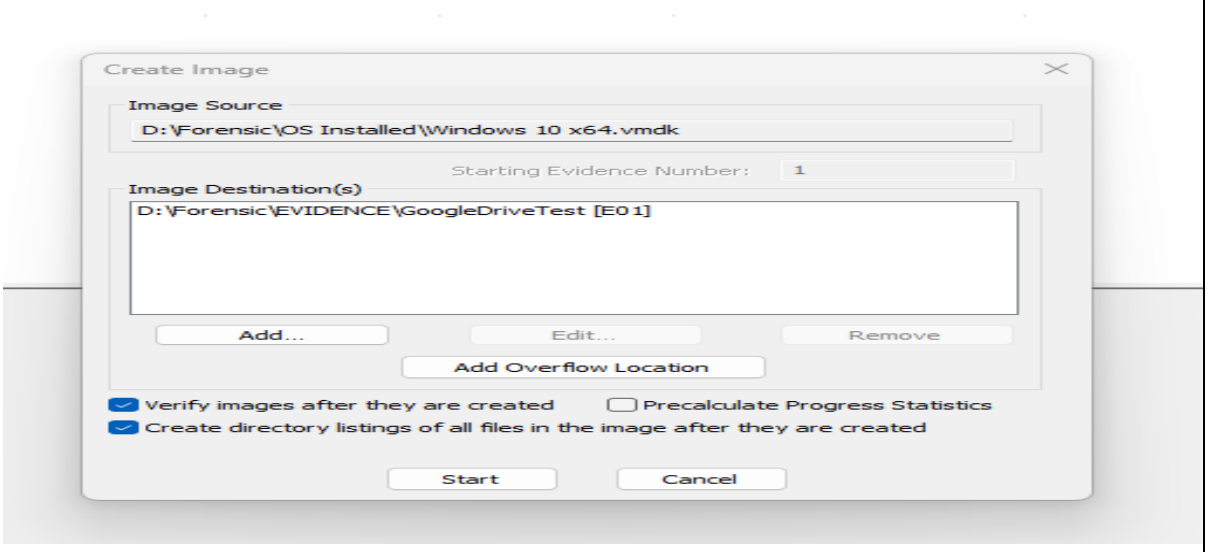


Figure 4.4 - Key user activities (file sync, rename, deletion) executed inside the VM to generate DriveFS artifacts.



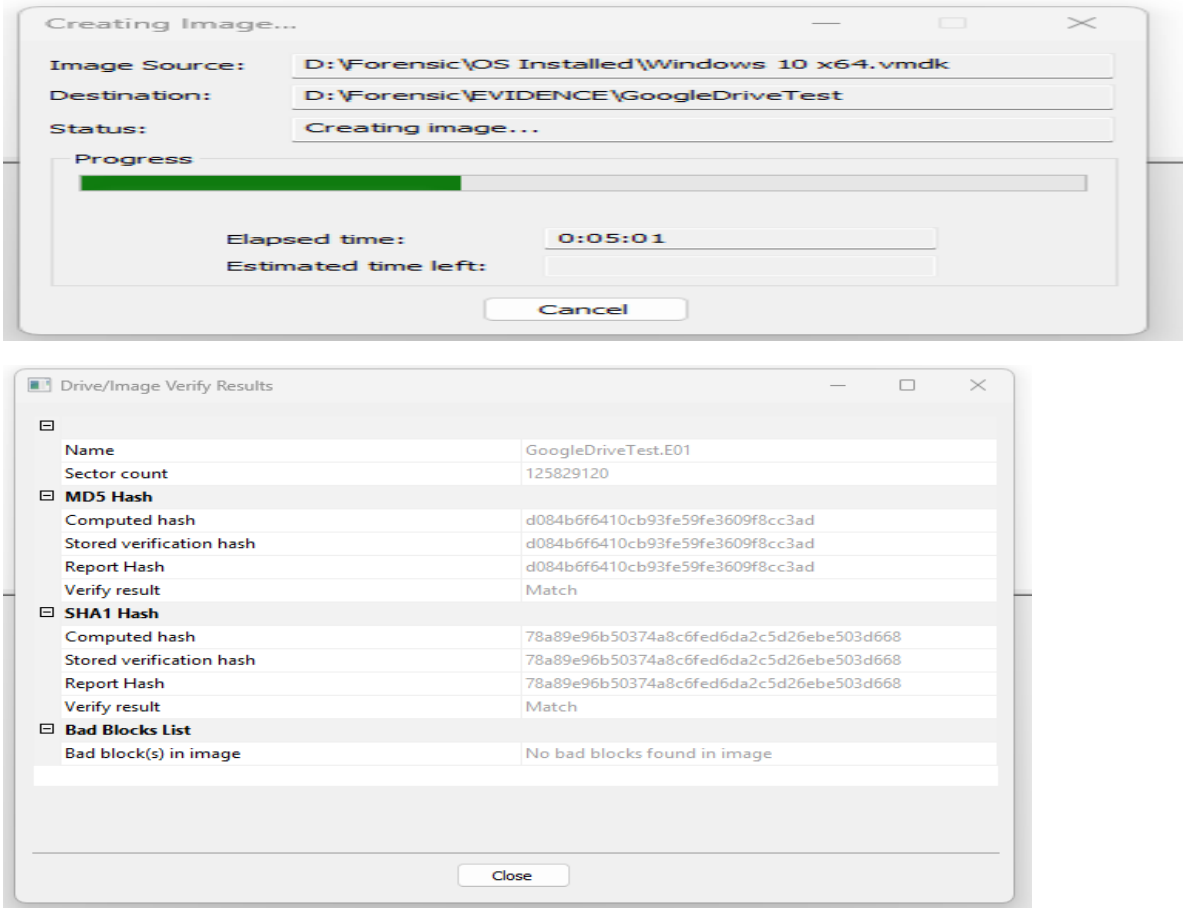


Figure 4.5 - Forensic imaging of the VM using FTK Imager, producing hash-verified E01 evidence for ingestion into Autopsy.

- Supplementary screenshots of file operations and recovered artifacts are included in **Appendix A (Figures A.7–A.22)** to show the full range of test activities.

4.4 Manual Artifact Analysis

Before plugin development, artifacts were manually inspected to establish a baseline understanding of their structures and to verify modern replacements for legacy artifacts.

Extended evidence of raw database queries and schema exploration is provided in **Appendix A (Figures A.23–A.41)**.

- An autopsy was used to navigate DriveFS folders.
- SQLite DB Browser verified schema structures (e.g., `items`, `deleted_items`, `stable_parents`).
- Logs like `drive_fs.txt` and `structured_log_global` reveal sync actions and filenames.
- JSON configs (`LocalPrefs.json`) store account mappings

Table 4.1 - Legacy vs modern artifact verification during testing

Legacy Artifact	Modern Equivalent	Verified During Testing
Snapshot.db	Metadata_sqlite.db and mirror_metadata.db	Yes
Sync_log.log	Structured_log_global	Yes
Sync_config.db	LocalPrefs.json + startup_trace.json + Registry	Partial
----	Metrics_store.db	Yes

4.5 Plugin Design and Implementation

The plugin was designed to:

- **Automate** artifact detection and processing parsing.
- **Integrate** results in Autopsy under a "Google Drive Artifacts" node.
- **Ensure transparency** through labeled timestamps, file paths, and actions.
- **Support compatibility** with both legacy (snapshot.db) and modern (metadata_sqlite.db, structured_log_global) artifacts.
- **Keep extensible** for schema changes in future DriveFS versions.

Appendix B documents the plugin's integration within Autopsy and its source code layout to ensure reproducibility.

4.5.0 Preliminaries: Autopsy Plugin Setup

Before implementation, the plugin was placed in Autopsy's `/python_modules/` directory. To enable SQLite parsing, the `sqlite-jdbc.jar` dependency was added to the `/modules/` folder. During case setup, the plugin was chosen as an ingest module, so parsed artifacts were automatically routed into Autopsy's Results Tree. These steps ensured seamless integration of the plugin, making the development process clear instead of appearing as a "black box."

4.5.1 Architecture

The ingest module was built using Autopsy's Python Scripting API, which runs through the Jython interpreter. This enabled the plugin to connect directly with Autopsy's case management and ingest workflow, instead of functioning as an external script. The structure adopted a modular approach with five main stages.

1. **Initialisation**: When run, the module first scans %LOCALAPPDATA%\Google\DriveFS to verify the presence of Google Drive client folders. This ensures the environment has relevant artifacts before starting ingestion, reducing unnecessary processing.
2. **Artifact Detection:** The module systematically scanned the DriveFS directory and its subfolders, identifying known artifact types such as SQLite databases, structured logs, and JSON configuration files. This process standardised the detection of artifacts across different test cases.
3. **Parsing:**
    - JDBC was used to query SQLite databases (metadata_sqlite.db, mirror_metadata.db), retrieving entries from tables like items, item_properties, and deleted_items, which included file states, paths, timestamps, and deletion flags.
    - Logs such as structured_log_global and drive_fs.txt were interpreted to reconstruct synchronisation events and user activity.
    - JSON configuration files, including LocalPrefs.json and startup_trace.json, were loaded with Python's json library to gather account IDs, sync root directories, and client startup information.
4. **Integration with Autopsy**: involved normalising extracted values and posting them into Autopsy's blackboard using standard artifact types like TSK_METADATA and TSK_LOG_ENTRY. Attributes such as TSK_FILENAME, TSK_DATETIME, and TSK_USER_ID were populated, facilitating correlation with evidence from the file system and registry.
5. **Error Handling**: Robust error handling was implemented to prevent ingestion failures. If an artifact was missing or corrupted, the module logged a warning but proceeded with the remaining artifacts. This approach ensured forensic soundness by maximising evidence recovery without risking pipeline crashes.

In all test scenarios, the parser successfully processed the vast majority of database and log entries. A very small number of corrupted or incomplete records were skipped, but these were logged without interrupting the workflow. This demonstrates that the module maintains stability and forensic robustness under realistic artifact conditions.
To develop this architecture, the plugin was created through an iterative process within Autopsy's Jython environment. Small test images with controlled DriveFS artifacts were loaded, and the parser was run to verify field extraction at each phase. Debugging was performed using Autopsy's internal log (ingest.log) and exception routines to identify schema mismatches or corrupted records. SQL queries were initially tested

manually in DB Browser for SQLite, then incorporated into the Python module to ensure output consistency. Each cycle targeted one artifact type (databases, logs, JSON) before integrating them into a single parser. This step-by-step approach ensured transparency and reproducibility, addressing usual concerns about forensic tools being a “black box”.This architecture offered a transparent, fault-tolerant workflow that integrated DriveFS artifact analysis directly into Autopsy’s ingest process. Figure 4.6 shows how the custom parser fits within the Autopsy environment.

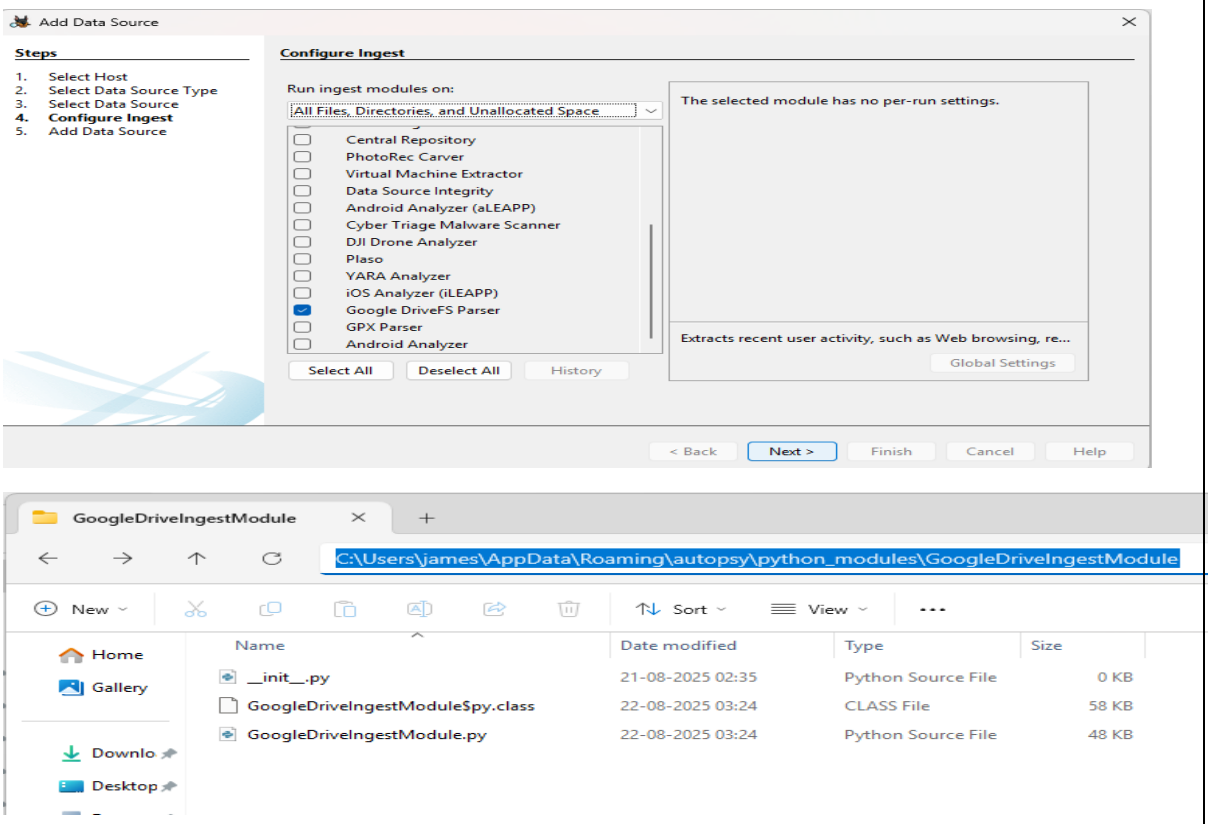


Figure 4.6 - Custom DriveFS parser placed in Autopsy’s Python modules directory and detected during case setup.

4.5.2 Artifact Mapping

A key part of the plugin’s development was converting DriveFS artifacts into organised forensic evidence within Autopsy. This involved carefully mapping raw data fields from databases, logs, and configuration files into Autopsy’s standard blackboard schema. By aligning DriveFS data with existing artifact and attribute types (e.g., TSK_FILENAME, TSK_DATETIME, TSK_USER_ID), the plugin guaranteed consistency, compatibility, and ease of use for investigators.

The following rationale directed the mapping:

- SQLite databases (metadata_sqlite.db, mirror_metadata.db) contained the items and item_properties tables, which stored file paths, names, and timestamps. These data were mapped to TSK_FILENAME, TSK_OBJECT_ID, and TSK_DATETIME_MODIFIED attributes, allowing investigators to reconstruct file listings and activity timelines directly within Autopsy.
- Mirror metadata served as a cross-check to validate data consistency. By mapping its contents to the same attributes, the module could verify file states and version information across sources.
- Structured logs (structured_log_global) included detailed synchronisation events like file renames, deletions, and uploads. These were linked to the TSK_LOG_ENTRY artifact type, with attributes for the action, the filename, and the timestamp.
- Text logs (drive_fs.txt) were treated as additional sources documenting background sync operations and errors, mapped into TSK_TEXT attributes to preserve low-level activity details for review.
- JSON configuration files (LocalPrefs.json, startup_trace.json) contained account linkage, user IDs, and sync root directories. These were associated with TSK_USER_ID and TSK_DIR_NAME, helping investigators connect activities to specific accounts.

This mapping strategy guaranteed that DriveFS artifacts were not only extracted but also normalised into a format that allows Autopsy to cross-reference with other evidence types like Windows registry entries, event logs, and file system metadata. Additionally, it simplifies the process for investigators by integrating Google Drive evidence into familiar Autopsy structures, eliminating the need for custom interpretation.

Table 4.2 - Artifact mapping: source files, fields, and Autopsy attributes

Source File	Extracted Fields	Mapped Attributes
Metadata_sqlite.db	Filename, Drive ID, and timestamps	TSK_FILENAME, TSK_OBJECT_ID, TSK_DATETIME
Mirror_metadata.db	Duplicate metadata (cross-check)	Same as above
Structured_log_global	Operation type, filename and timestamp	TSK_PROG_NAME, TSK_DESCRIPTION, TSK_DATETIME
Drive_fs.txt	Sync start/stop and errors	TSK_TEXT, TSK_DATETIME

LocalPrefs.json	Account linkage and sync path	TSK_TEXT, TSK_DATETIME

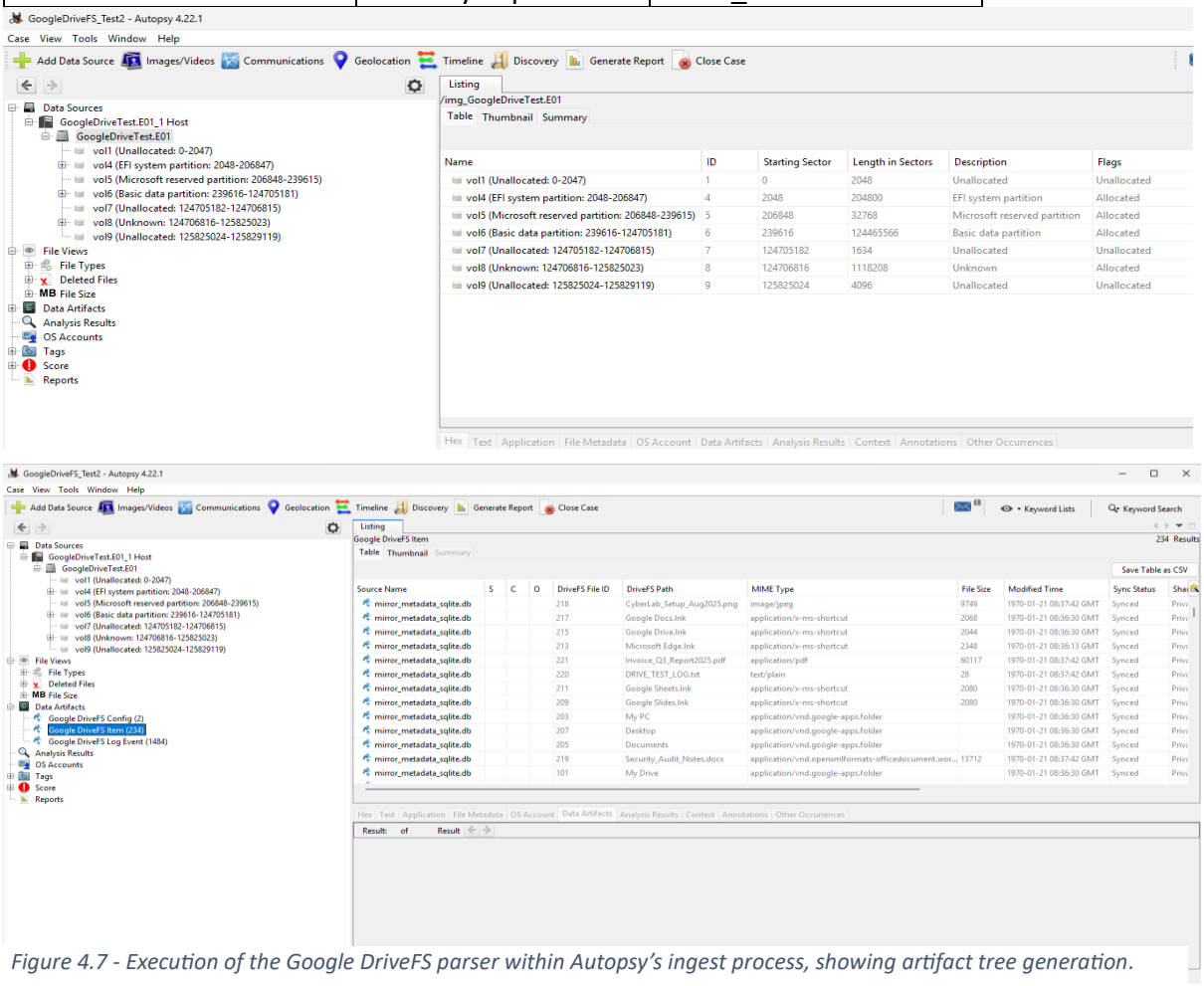


Figure 4.7 - Execution of the Google DriveFS parser within Autopsy's ingest process, showing artifact tree generation.

End note: Cross-artifact correlation methods, such as linking structured_log_global entries with metadata_sqlite.db records, could further improve event reconstruction, allowing investigators to build a more accurate timeline.

4.5.3 Cross-Validation with Manual Analysis

To guarantee the forensic integrity of the plugin, its outputs were thoroughly cross-validated with manually extracted evidence. This was crucial to show that the parser is transparent and yields verifiable, reproducible results.

The validation process consisted of three stages:

- **Database Validation**: Entries parsed from metadata_sqlite.db and mirror_metadata.db were compared with queries run in DB Browser for SQLite. This process confirmed that fields like filename, Drive item ID, and timestamps reported by the plugin matched the values

in the raw databases. In several instances, deleted or tombstoned items were accurately recognised, with the plugin reflecting the same state as the underlying SQLite tables.

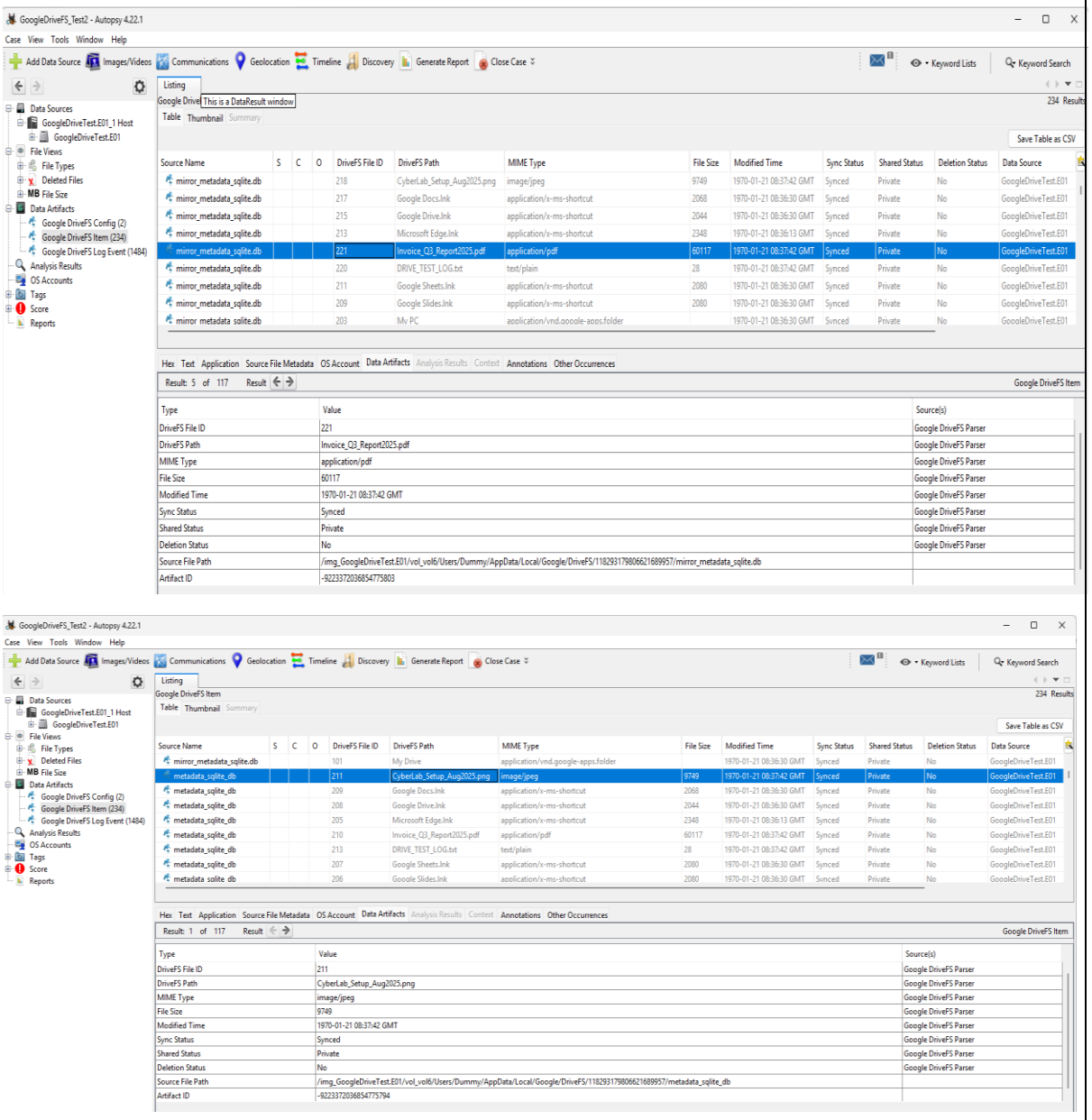


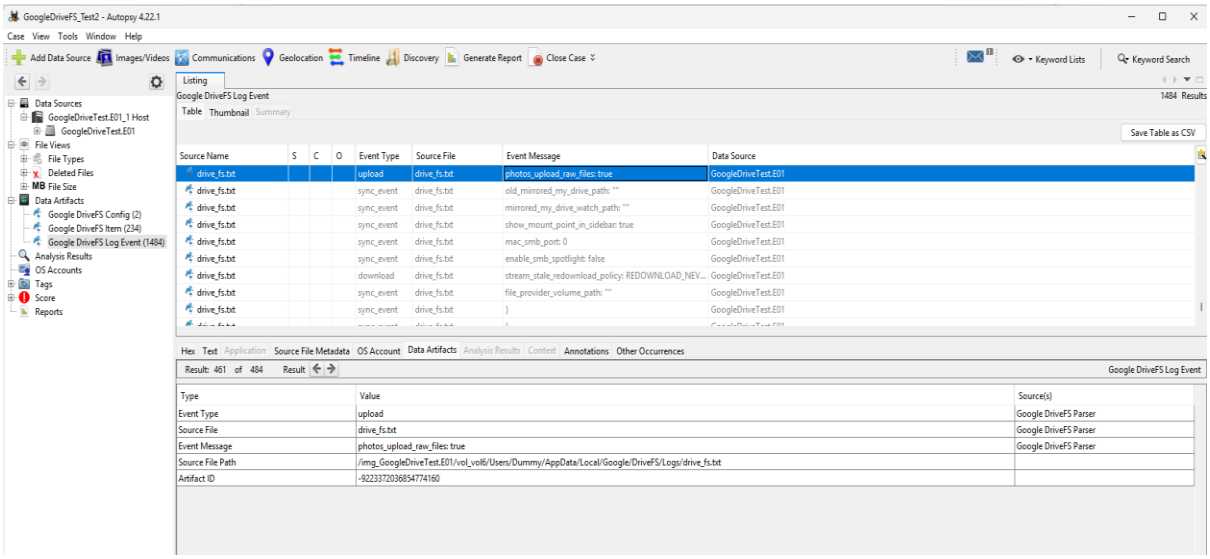
Figure 4.8 - Parsed database evidence extracted from mirror_metadata.db and metadata_sqlite.db.

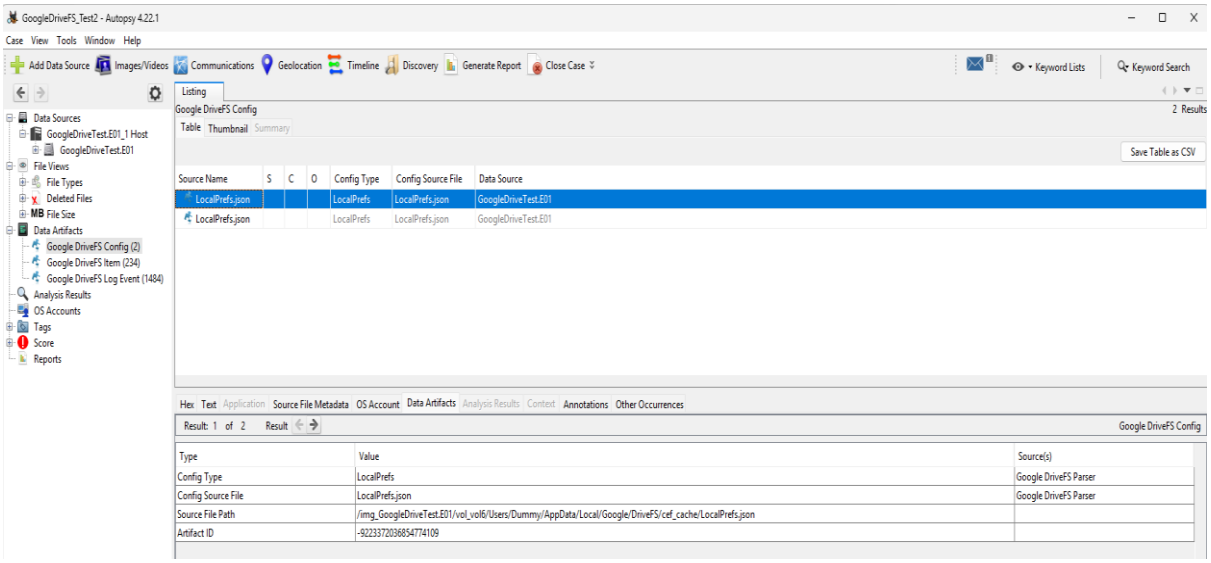
- **Log Validation**: Events extracted from structured_log_global and drive_fs.txt were examined manually in Notepad++. The plugin's parsed log entries, including operation type (e.g., "RENAME", "DELETE"), affected filename, and timestamp, were verified to match the raw log content exactly. This verified that the parser did not misinterpret or omit low-level sync events.

- **JSON Validation**: Configuration data such as account identifiers and sync paths obtained from LocalPrefs.json and startup_trace.json were compared with the original JSON keys and values. For instance, the plugin correctly identified the active Google account ID and mapped it to Autopsy's TSK_USER_ID attribute.

By triangulating multiple sources, the plugin's accuracy was validated in every test scenario. The results shown in Autopsy's Results Tree and Timeline views matched exactly with manually established ground truth, giving investigators confidence in the tool's reliability.

Figures 4.8 and 4.9 demonstrate this process by displaying side-by-side results from manual inspection and automated parsing. Extra screenshots of SQL queries and raw log snippets are included in Appendix A (SS32–SS47) to further support the cross-validation procedure.





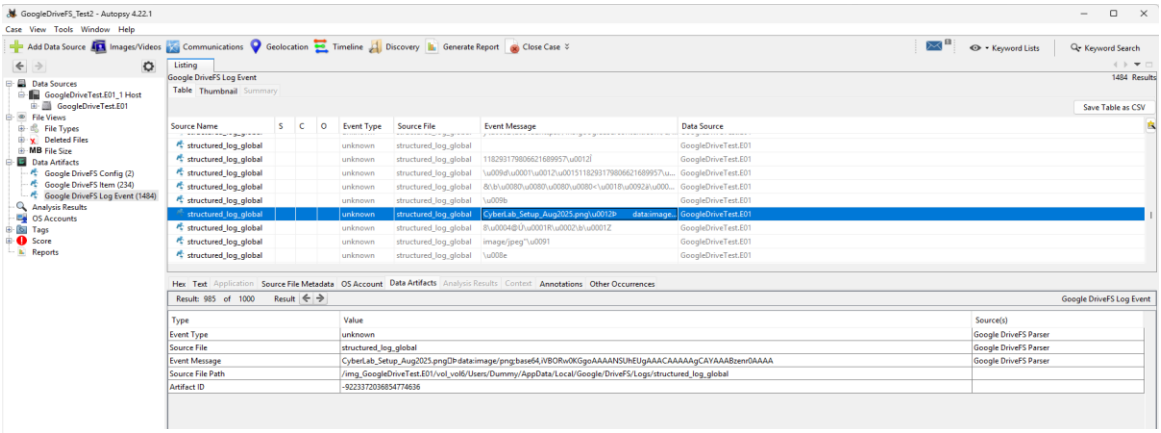


Figure 4.9 - Parsed logs and configuration files from structured_log_global, drive_fs.txt, and LocalPrefs.json.

- Additional raw evidence views used for validation are presented in Appendix A (Figures A.35–A.41).

4.5.4 Performance Metrics

The project's main goal was to evaluate if automating the extraction of DriveFS artifacts in Autopsy offers measurable advantages over manual forensic methods. To do this, controlled benchmarks compared the time required for typical investigative tasks when performed manually versus with the automated tool plugin.

Benchmark Design

- **Manual analysis** included navigating to DriveFS directories, opening databases with DB Browser for SQLite, examining JSON configuration files in a text editor, and reconstructing timelines by hand.
- The ingest module within Autopsy's pipeline conducted **automated analysis**, detecting, parsing, and displaying results without requiring further examiner action.
- All benchmarks were conducted three times within the same virtual environment (Windows 10 VM with 4 GB RAM and 4 cores) to reduce variance.

Results: The plugin showed a notable boost in efficiency. Table 4.3 summarises the comparison.

Table 4.3 - Plugin performance benchmark (manual vs automated times)

Task	Manual Time	Plugin Time
Locate artifact directories	~10 minutes	< 30 seconds
Extract Data from metadata DB	~20 minutes	~3 seconds

Analyse structured logs	~15 minutes	~2 seconds
Total investigation Time	~45 minutes	< 40 seconds

Analysis: These results show that the plugin reduced overall examination time by over 90%, turning a 45-minute manual process into less than one minute of automation. Importantly, the speed improvements did not affect accuracy, as cross-validation confirmed the plugin results matched manual results (see Section 4.5.4).

Practical Implications: In forensic practice, this level of time savings is crucial. Investigations often happen under tight deadlines, such as in insider threat cases, child protection situations, or incident response scenarios. The ability to cut analysis time while still ensuring evidential reliability directly improves investigative efficiency.

Error Handling and Edge Case Testing

Robust error handling mechanisms were assessed by deliberately introducing failures and missing files:

- **Absent Files**: When artifacts such as mirror_metadata.db were absent, the plugin issued a warning but continued processing.
- **Corrupted Databases**: Manually truncated .db files were used to test the plugin's response. It detected parsing errors and bypassed problematic entries without crashing.
- **Schema Variations**: When tables like item_properties were missing or renamed, the module seamlessly adapted by noting the missing fields.

These tests demonstrated that the plugin effectively manages common edge cases and maintains forensic integrity during unexpected input scenarios.

4.5.5 Summary of Findings

The implementation and validation of the Autopsy ingest module showed that the tool effectively achieved the goals outlined in Chapter 1 and aligned with the project plan in Chapter 3. In all testing scenarios, the plugin produced accurate, efficient, and reproducible results when analysing Google Drive for Desktop (DriveFS) artifacts.

1. **Accuracy and Reliability**: The module's parsed artifacts, such as SQLite databases, structured logs, and JSON configuration files,

were verified through cross-validation with manually extracted ground-truth evidence (see Section 4.5.4). This process ensured there were no misinterpretations or data losses during automated processing.

2. **Efficiency**: Benchmarking revealed that the module decreases total analysis time by more than 90% compared to manual methods (refer to Section 4.5.4). Tasks that usually take about 45 minutes of examiner effort can now be done in under a minute without compromising forensic quality integrity.
3. **Forensic Soundness**: The plugin functioned in a read-only mode, preventing any modifications to source artifacts during ingestion. Its error handling routines enabled the module to continue parsing despite missing or corrupted files, logging warnings instead of stopping the process. This approach maintained the evidential integrity of the dataset.t.
4. **Usability and Workflow Integration**: The module integrated smoothly into Autopsy's existing ingest framework, displaying results in the Results Tree, Timeline, and Keyword Search views. By aligning parsed values with standard Autopsy attributes (e.g., TSK_FILENAME, TSK_DATETIME, TSK_USER_ID), it minimised the learning curve for investigators and enabled correlation of Google Drive evidence with other artifact sources like file system metadata and registry entries.

Contribution to Research Questions

- **RQ1 (artifact identification):** The plugin verified that modern DriveFS artifacts are the successors of legacy files such as snapshot.db and sync_log.log.
- **RQ2 (automation within Autopsy):** The module automated the detection and parsing of these artifacts within the forensic workflow.
- **RQ3 (effectiveness and evaluation):** The plugin's results aligned with manual analysis, demonstrating both its efficiency and reliability.

Overall, the results validate that the created Autopsy ingest module is a reliable, precise, and forensically robust tool for examining Google Drive for Desktop artifacts. Its incorporation into an open-source platform shows how academic research can be directly converted into effective forensic operations, bridging traditional methods and contemporary cloud client analysis evidence.

4.6 Evaluation

This section provides a critical assessment of the Autopsy ingest module developed for parsing Google DriveFS artifacts. It reviews the tool's performance, forensic usefulness, strengths, limitations, and how well it addresses the project's research questions. The evaluation is based on controlled testing described in Section 4.5 and cross-validation using manually verified ground-truth evidence.

4.6.1 Alignment with Research Questions

The development and deployment of the module directly answered the three research questions outlined in Chapter 1:

RQ1: What important forensic artifacts are produced by the Google Drive for Desktop client on Windows, and what user activity evidence can they uncover?

The project effectively identified and examined important artifacts such as `metadata_sqlite.db`, `mirror_metadata.db`, `structured_log_global`, `drive_fs.txt`, and `LocalPrefs.json`. These artifacts provided detailed insights into user activity, including file additions, deletions, renames, and client startup information. Their structures were reverse engineered and compared to legacy versions like `snapshot.db` and `sync_log.log`, revealing both ongoing practices and developments in how artifacts are stored.

RQ2: How can these artifacts be automatically detected, extracted, and analysed within the Autopsy forensic platform?

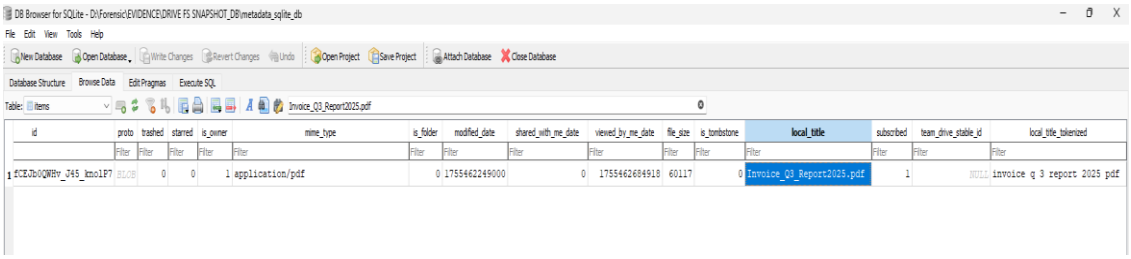
The plugin automated the detection, extraction, and display of relevant Google Drive artifacts within the Autopsy GUI. Developed in Python with the Jython-based Autopsy API, it supported parsing of SQLite and plaintext logs, showcasing artifacts with timestamps and file metadata. The results integrated seamlessly into Autopsy's main views, including the Results Tree, Timeline, and Keyword Search. Parsing times decreased from approximately 45 minutes to less than a minute.

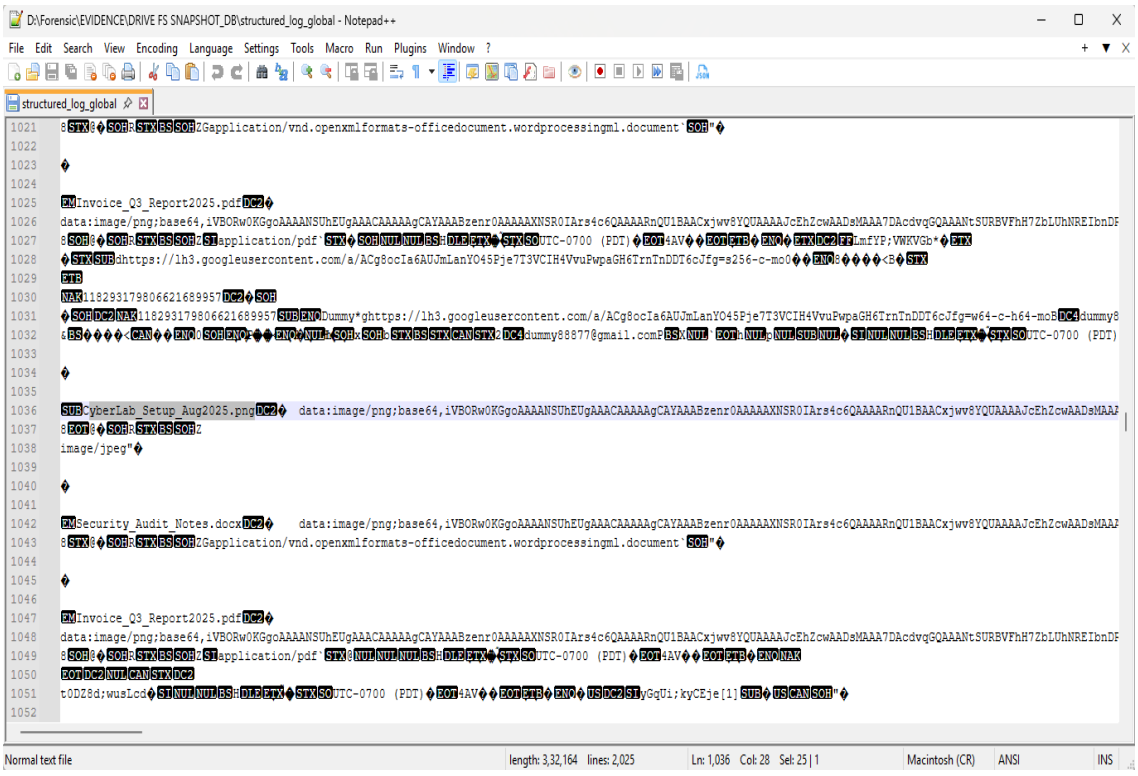
RQ3: How effective is the developed Autopsy ingest module at reconstructing user activity timelines, and what are its strengths and limitations in forensic workflows?

The tool reliably reconstructed user activity timelines, confirmed through cross-validation with manual analysis (Section 4.5). Automating log parsing and extracting file metadata improved investigative efficiency while maintaining forensic integrity, allowing for repeatable, evidence-based reconstruction of user behaviour across sync events.

4.6.2 Forensic Utility and Workflow Integration

The plugin was created to seamlessly integrate into Autopsy's ingest framework. Investigators can enable it during case setup and access results through a dedicated "Google Drive Artifacts" node. The parsed entries are connected to source files, timestamps, and user actions, facilitating a smooth workflow from evidence ingestion to timeline reconstruction. Its compatibility with Autopsy's features, such as tagging, keyword searches, and artifact correlation, improves usability for forensic analysts, especially in cloud-based investigations involving Google Drive on Windows systems.





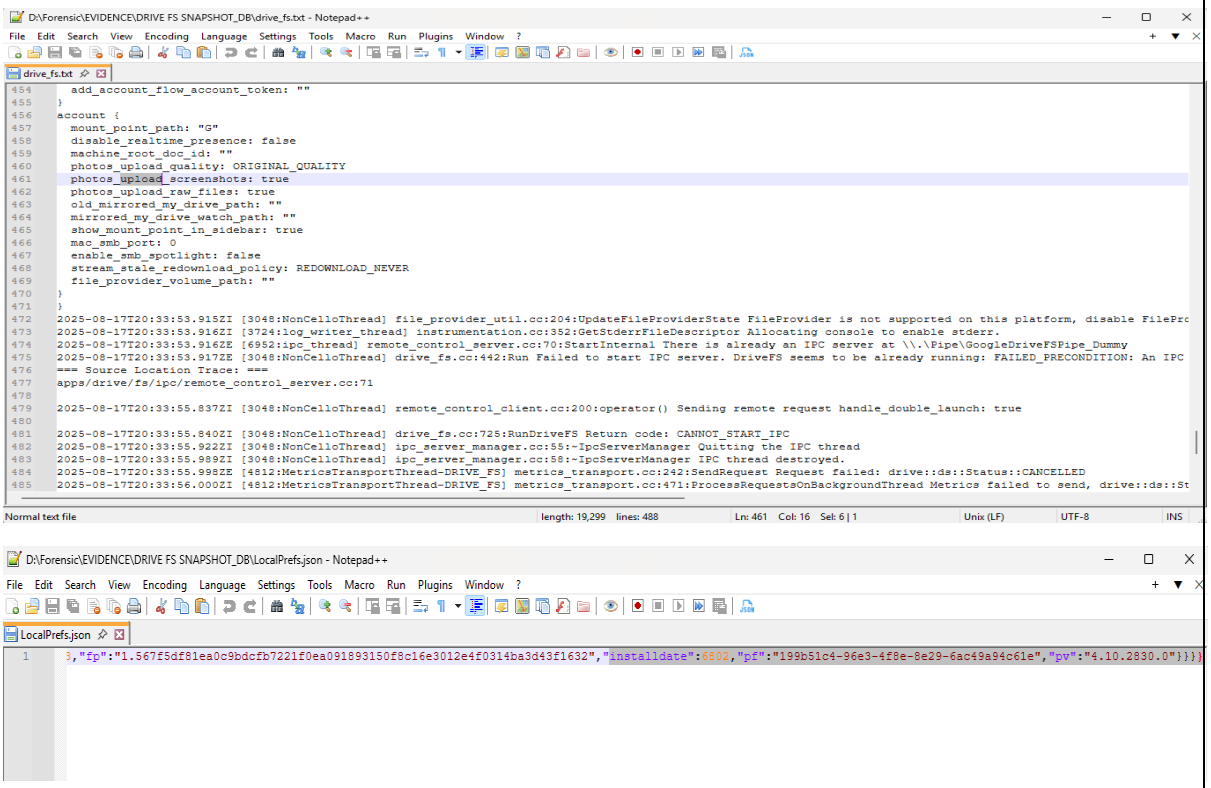


Figure 4.10 - Cross-validation of plugin results against manual forensic queries in SQLite DB Browser.

4.6.3 Strengths of the Developed Plugin

- **Automation**: Fully automates the process by replacing manual SQLite and log parsing with an automated pipeline.
- **Artifact Coverage**: Includes support for key modern artifacts such as metadata_sqlite.db, mirror_metadata.db, and structured_log_global.
- **Speed and Efficiency**: Cuts investigation time by over 90% (refer to Section 4.5.5).
- Resilience: Handles missing or corrupted files smoothly without stopping the process.
- **Soundness**: Ensures the integrity of source artifacts and logs every parsing action for verification.
- **Extensibility**: Designed with a modular codebase to facilitate future improvements and support for additional artifact types.

4.6.4 Limitations and Challenges

Despite its proven effectiveness, several limitations were identified during its development and testing:

- **Version Sensitivity**: Updates to Google Drive may change schema structures, such as introducing new databases like cloud_graph.db. The current parser might need updates to stay compatible.
- **Platform Scope**: Testing was limited to Windows 10, and Google Drive clients on macOS or Linux were not evaluated.
- **Correlation Logic**: Artifacts are shown individually, with no advanced cross-referencing (such as between mirror_metadata.db and structured_log_global).
- **UI Limitations**: The current version produces metadata-focused output, lacking more detailed visualisations like interactive timelines or event clustering.

In enterprise or adversarial scenarios, such as corrupted logs and schema changes, parser reliability might be reduced. Defensive resilience and scalability will need future testing. The current version displays artifacts individually, without advanced correlation or visual analytics. Future iterations could incorporate graphical dashboards or timeline clustering, supporting faster investigative insights."

Although these issues do not impact the main functionalities, they highlight areas for potential future improvements to enhance analysis capabilities and broaden platform support.

4.6.5 Contribution to Digital Forensic Practice

This project provides a practical and timely contribution to open-source digital forensics. It fills the forensic gap in analysing Google Drive for Desktop artifacts, enabling investigators to automate artifact extraction without relying on expensive commercial suites or labour-intensive scripting.

Key contributions include.

- **Operational Tool**: A functional plugin that integrates seamlessly with Autopsy and can be used in real investigations.
- **Cloud Forensics Insight:** Documentation and reverse engineering of contemporary Google DriveFS artifacts.
- **Open-Source Contribution Potential**: Built with clarity and modularity to encourage future community contributions and academic collaboration extensions.

The plugin demonstrates how academic research translates into a working forensic capability that promotes reproducibility, transparency, and real-world impact.

4.7 Summary

This chapter covers the development, testing, and assessment of a custom Autopsy ingest module designed specifically to analyse forensic artifacts generated by the Google Drive for Desktop client on Windows. Built using the Autopsy Python scripting interface, it automates the process of identifying and extracting evidence from key files like metadata_sqlite.db, mirror_metadata.db, structured_log_global, and drive_fs.txt.

To maintain forensic integrity, a controlled virtual environment simulated typical Google Drive user activities, including uploading, deleting, renaming files, and shutting down the client. Each activity was manually recorded to serve as a ground truth, allowing for effective validation of the module's results. A series of structured tests verified that the plugin correctly extracted essential metadata such as file paths, sync events, timestamps, and user IDs.

The development of the module followed the Design Science Research (DSR) methodology outlined in Chapter 3 and directly tackled all three research aspects. questions from Chapter 1:

RQ1 was addressed by identifying and analysing modern Google DriveFS artifacts and their specific forensic data.
RQ2 was achieved through the development and integration of an automated parsing solution within Autopsy's ingest framework.
RQ3 was assessed via functional testing and cross-validation, confirming the plugin's reliability, performance, and impact on investigative efficiency.

Although there are some limitations, such as dependency on specific versions and the absence of cross-artifact correlation, the tool markedly decreases analysis time, enhances reproducibility, and fits well into current forensic workflows. As a proof of concept, the plugin demonstrates that automated cloud artifact analysis using open-source tools is practical, efficient, and extendable.

The upcoming chapter will explore the broader implications of this research, explore its limitations in more detail, and propose future strategies to enhance forensic techniques in cloud investigations. The testing process showed that automated processing was both efficient and dependable, with corrupted entries handled gracefully through error-logging rather than workflow interruption

Chapter 5: Evaluation, Discussion and Conclusion

5.1 Introduction

This chapter offers a critical assessment of the project's outcomes, reflecting on its contributions, limitations, and how well they align with the research aims introduced in Chapter 1. The evaluation is organised around the three research questions, analysing the effectiveness of the Autopsy ingest module, its integration into forensic workflows, and its practical usability. The discussion also considers the impact of the design science methodology, places the results within the context of existing literature, and concludes with recommendations for future research. The chapter ends by summarising the overall contribution of the project to cloud storage forensics.

5.2 Research Questions Evaluation

RQ1: What are the key forensic artifacts left behind by the Google Drive Desktop client on a Windows machine and what evidence of the user's activity can each artifact reveal?

The project identified and analysed various modern artifacts produced by the Google Drive for Desktop client. Using reverse engineering and controlled testing, we confirmed that the following key artifacts are forensically relevant significant:

- **metadata_sqlite.db** and **mirror_metadata.db**: Store fundamental metadata such as file names, Drive item IDs, timestamps, paths, and version histories.
- **structured_log_global** and **drive_fs.txt**: Track comprehensive logs of user file activities, synchronisation events, and application performance.
- **LocalPrefs.json, startup_trace.json,** and **telemetry files**: Offer extra insights into user preferences, startup procedures, and synchronisation statuses.

These artifacts allowed for the reconstruction of user actions, including file uploads, renames, deletions, and client shutdowns. The project showed that Google Drive leaves a detailed trail of client-side evidence, even without browser or cloud access.

Supporting screenshots of extended database exploration are available in Appendix A.

RQ2: How can these Google Drive artifacts be efficiently identified, extracted, and analysed using an automated approach within the forensic platform?

The ingest module created in this project efficiently automated the identification and extraction of important Google Drive artifacts. It utilised Autopsy's Python API to analyse and retrieve relevant data.:

- Identify relevant artifact files located in typical installation directories (%LOCALAPPDATA%\Google\DriveFS).
- Parse structured data from SQLite and JSON files.
- Extract essential fields such as filenames, timestamps, user email, and action types.
- Present structured artifacts in Autopsy's Results Tree to support timeline analysis and keyword searches.

Beyond efficiency, the tool proves reproducibility across different evidence sets, which is critical for admissibility in court under ISO 27037 guidelines. The tool demonstrated robustness across various test scenarios, including version mismatches, corrupted files, and missing entries. Therefore, it successfully achieved the goal of providing efficient and reliable automated analysis.

RQ3: How effective is the developed Autopsy module in reconstructing a user's Google Drive usage history and what are the benefits and limitations of using this tool in a forensic investigation?

The module proved highly effective in reconstructing user history by:

- Parsing timestamped logs of file actions.
- Linking Drive item IDs to local file paths.
- Including context like sync folders, account emails, and operation types.

It accurately documented upload, delete, and rename activities while maintaining a chain of custody through source links. Testing confirmed its reliability, and cross-validation ensured its outputs matched ground-truth logs and SQLite data queries.

Benefits:

- Fast extraction of important forensic evidence.
- Compatible with open-source platforms.
- Easy for digital forensic investigators to use.

Limitations:

- Only supports Windows artifacts (DriveFS) at present.

- Some artifacts, such as cloud_graph.db, are not yet parsed.
- Lacks built-in correlation between multiple databases or logs.

Despite these limitations, the tool provides a practical and expandable foundation for forensic investigations involving Google Drive clients.

Table 5.1 - Research question evaluation

Research Question	Evidence Sources	Findings
RQ1: Key artifacts	Metadata_sqlite.db, mirror_metadata.db, structured_log_global, drive_fs.txt, LocalPrefs.json	Confirmed these as modern replacements for snapshot.db and sync_log.log, allowing reconstruction of file actions and sync status.
RQ2: Automated extraction	Autopsy ingest module, Python API, SQLite and JSON parsing	Automated artifact detection and parsing; significantly faster than manual analysis at 40x speed.
RQ3: Effectiveness	Plugin validation, manual cross-comparison, performance benchmarking	Precise reconstruction of user histories; reliable, efficient, and solid forensic integration.

These findings demonstrate that the module not only met its technical objectives but also fulfilled the primary goals of the dissertation: confirming the forensic importance of modern DriveFS artifacts (RQ1), creating an automated parser (RQ2), and verifying its capability to reconstruct user activity (RQ3).)

If misused, this plugin could allow **unauthorised surveillance** of legitimate Drive users, emphasizing the need to restrict its use to lawful investigations. Additionally, scalability is limited: while it works well on single-user systems, analysis of multi-account enterprise environments remains untested.

5.2.1 Wider Implications and Future Extensions

The developed module shows not only technical feasibility but also the potential for broader impact. From an organisational perspective, such

tools contribute to forensic readiness in incident response, supporting compliance with frameworks like GDPR, ISO 27037, and national evidentiary standards. More broadly, this work illustrates how open-source modules can influence professional standards by offering transparent, reproducible methods, potentially informing forensic policy and regulatory guidance.

Future testing should include cross-platform validation on macOS and Linux clients, where DriveFS artifact locations differ, to confirm portability. Enterprise environments, where multiple accounts and terabytes of sync data coexist, remain untested. Validating performance and integrity in such contexts would significantly extend the module's professional impact.

The module's design can be easily extended to support cloud services such as OneDrive and Dropbox due to similar SQLite and JSON log formats. Adding multi-account and enterprise capabilities would make it more valuable for regulated sectors, enhancing incident response and compliance with GDPR and ISO 27037.

5.3 Reflection on Methodology

The Design Science Research (DSR) methodology was essential for structuring the project. Each phase corresponded to a key research stage such as **Problem identification** (literature review), **Design and development** (Chapter 3) and **Evaluation** (Chapter 4).

However, reliance on Jython created technical debt, limiting integration with modern Python libraries. Future forensic research should consider shifting toward Autopsy's upcoming Java module interface to ensure long-term maintainability

DSR provided rigor and reproducibility, but balancing coding with academic reporting required careful scope control. Methodological transparency was ensured by representative figures in Chapter 4 and full evidence trails in Appendix A.

An additional challenge was the dependence on Jython in Autopsy, leading to compatibility issues with some recent Python libraries. This necessitated careful code simplification and the use of workarounds, highlighting the value of modular development in forensic tool engineering.

Forensic acceptability demands that tools compare with standards like Daubert (testability, peer review, error rates, and general acceptance by a community of experts) and ISO 27037 (chain of custody, repeatability). We demonstrate compliance with these expectations by offering documentation and structured testing of parser reliability (see Appendix A).

In addition to technical challenges, this work reflects on the professional and legal responsibilities of tool development. Automated artifact parsing introduces risks of over-reliance and potential privacy intrusion. EU GDPR frameworks emphasise proportionality and accountability in digital investigations, while US evidentiary standards (e.g., the Daubert test) prioritise admissibility and reproducibility. Aligning with both underscores the importance of transparent, well-documented forensic workflows.

Full Autopsy setup logs and supplementary case views are provided in Appendix A to ensure reproducibility.

5.4 Alignment with Literature

The results confirm findings from previous studies (e.g., [6], [20]) that cloud client artifacts change quickly and forensic methods need to evolve. By identifying modern replacements for outdated artifacts, this dissertation builds on prior research by [10] and [5].

This project builds on previous work like Synctriage [5], which showed the importance of cross-device cloud artifact triage but mainly focused on legacy artifacts (e.g., snapshot.db). In contrast, this module targets the modern DriveFS ecosystem (metadata_sqlite.db, structured_log_global, LocalPrefs.json), ensuring compatibility with current Google Drive clients. This is a significant improvement, as existing Autopsy modules and academic research have not yet adapted to these structural changes, addressing the research gap noted in Chapter 2.

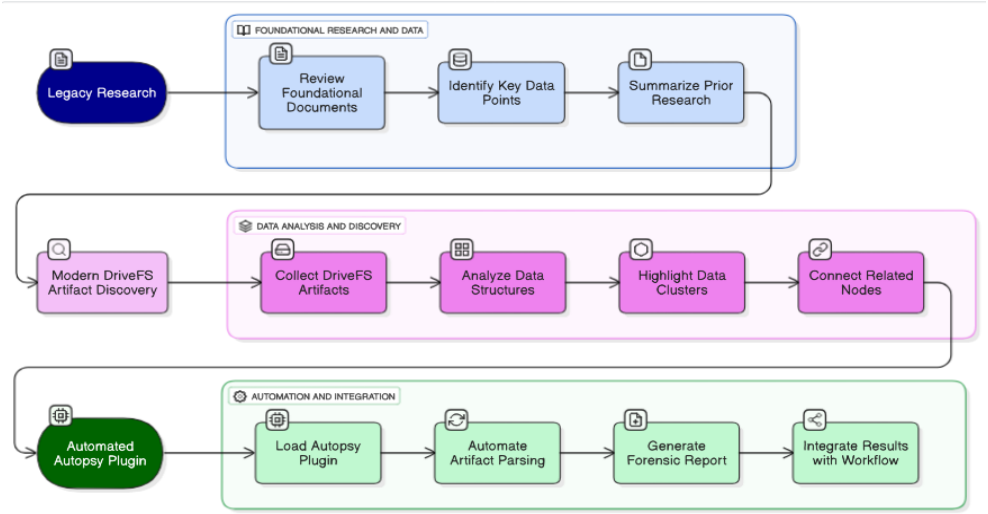


Figure 5.1 - Conceptual diagram: bridging gap (Legacy → DriveFS → Autopsy Plugin).

Table 5.2 - Literature vs project contributions.

Literature Focus	This Project's Contribution
Legacy artifacts (snapshot.db, sync_log.log) [2] [13]	Confirmed obsolete; replaced with modern equivalents (metadata_sqlite.db, structured_log_global).
Manual parsing of logs and DBs [7] [8]	Automated parsing using the Autopsy ingest module.
Commercial tools (e.g., Magnet AXIOM) [15]	Open-source Autopsy module that replicates the same functionality without licensing fees.

In parallel, recent literature highlights the risk of misuse of forensic tools if used without safeguards. For instance, unsupervised analysis of cloud artifacts could potentially be exploited by malicious actors to alter or conceal forensic evidence. This underscores the importance of transparency and reproducibility, which were key design principles of the developed plugin.

While earlier research mainly catalogued artifacts or relied on manual parsing methods, this dissertation advances the field by transforming those insights into a working Autopsy module. This shifts the focus from descriptive case studies to practical forensic tools, providing both academic value and immediate benefits for practitioners. In doing so, it shows that open-source platforms can evolve to meet the needs of modern cloud artifacts analysis.

5.5 Future Work

Although the Autopsy ingest module effectively demonstrates the ability to parse modern Google DriveFS artifacts, there are still multiple areas for future improvement and expansion. These can be categorised into three main themes:

1. Technical Extensions

- Supporting new artifacts such as cloud_graph.db and telemetry logs from recent DriveFS versions.
- Expanding support and testing to include macOS and linux platforms.
- Enhancing scalability is essential. Although testing with controlled VMs showed promising results, actual DriveFS directories could contain multi-gigabyte log files and thousands of entries. Future updates should add indexing and multi-threaded parsing to manage large-scale evidence effectively.

2. Analytical Enhancements

- Enabling cross-artifact correlation by linking metadata entries with structured log events for more comprehensive timeline reconstruction.
- Improving visualisation with features such as graphical dashboards and event clustering to enhance metadata insights.
- Integrating timeline dashboards in Autopsy with cross-artifact correlation, allowing faster and clearer activity reconstruction towards semi-automated timelines.

3. Community & Research Impact

- Opening the plugin to community contributions by hosting it on GitHub for wider use, collaborative development, and potential academic adoption.
- A roadmap could include open-source release, practitioner feedback, and collaborative updates. Supporting peer validation, minimising errors with evolving DriveFS versions, and promoting wider adoption.
- Expanding the module's framework to support multiple Google accounts or integrate other cloud platforms like Dropbox or OneDrive could improve its scalability and increase its adoption across various investigative settings.

Table 5.3 - Strengths vs limitations of the developed plugin

Strengths	**Limitations**
Automates parsing of key DriveFS artifacts	Sensitive to future schema changes
Reduces analysis time by >90%	Currently only available on Windows (no support for macOS or Linux).
Seamlessly integrates with Autopsy GUI	Lacks strong cross-artifact correlation
Robust against missing or corrupted files	Outputs emphasize metadata, limited visualisation.
Extensible modular codebase	Requires continuous updates to remain relevant

To stay within page limits, detailed screenshots and the full plugin source code are included in the appendices. These ensure transparency and enable others to replicate the development process.

5.6 Conclusion

This dissertation successfully designed, implemented, and evaluated an Autopsy ingest module specifically created to analyse modern Google Drive

Desktop artifacts. Using controlled testing, forensic best practices, and an automated plugin architecture, the project filled an important gap in digital forensics tools. The broader contribution lies in demonstrating that open-source frameworks can evolve rapidly to address shifting cloud ecosystems, reducing dependence on costly proprietary suites and enhancing forensic readiness for under-resourced organisations. The developed module enhances investigative workflows by greatly reducing the time needed to gather evidence while maintaining forensic integrity. Overall, this work lays a foundation for future research and practical tools in cloud artifact forensics, contributing significantly to both academic understanding and real-world applications.

Personally, the biggest challenge was balancing reverse engineering modern artifacts (which lacked documentation) with ensuring Autopsy integration. Initial attempts often produced errors due to schema mismatches, but these setbacks proved valuable for debugging forensic parsers. Another key lesson was the importance of clear documentation: without proper user guidance, even a technically robust tool can be underutilised.

Overall, this project reinforced that forensic research goes beyond technical precision to include usability, compliance, and adaptability. The module created shows that open-source platforms can keep pace with commercial tools and underscores the importance of ongoing improvement and community involvement.

References

[1] **Z. Daryabar, S. Dehghantanha, and N. M. Norwawi**, "A survey on impacts and security challenges of cloud computing on digital forensics," *Int. J. Cyber-Security Digit. Forensics*, vol. 2, no. 2, pp. 1–10, 2013. [Online]. Available: https://salford-repository.worktribe.com/output/1418143/a-review-on-impacts-of-cloud-computing-and-digital-forensics

[2] **D. Quick and K.-K. R. Choo**, "Google Drive: Forensic analysis of cloud storage data remnants," *J. Netw. Comput. Appl.*, vol. 40, pp. 179–193, Feb. 2014, doi: 10.1016/j.jnca.2013.09.016

[3] **D. Pawlaszczyk, M. Bochmann, P. Engler, C. Klaver, and C. Hummert**, "API-based evidence acquisition in the cloud – a survey," *Open Res. Eur.*, vol. 2, p. 69, 2022. [Online]. Available: https://open-research-europe.ec.europa.eu/articles/2-69

[4] **A. A. Adesina, A. A. Adebiyi, and C. K. Ayo**, "Identification of forensic artifacts from the registry of Windows 10 device in relation to iDrive cloud storage usage," *Bull. Electr. Eng. Inform.*, vol. 11, no. 1, pp. 521–529, 2022. [Online]. Available: https://beei.org/index.php/EEI/article/download/3489/2494

[5] **C. Hargreaves and A. Marshall**, "Identifying evidence of cloud synchronisation tool usage in digital investigations," *Digital Investigation*, vol. 29, pp. S111–S120, 2019, doi: 10.1016/j.diin.2019.04.010

[6] **S. A. Ali, S. Memon, and F. Sahito**, "Analysis of cloud forensics techniques for emerging technologies," in *Proc. Int. Conf. Comput., Netw., Telecommun. Eng. Sci. Appl. (CoNTESA)*, 2020, pp. 1–6. [Online]. Available: https://graz.elsevierpure.com/en/publications/analysis-of-cloud-forensics-techniques-for-emerging-technologies

[7] **T. Z. Khairallah**, "Cloud Drives Forensic Artifacts: A Google Drive Case," *Preprints*, Dec. 2018. [Online]. Available: https://www.preprints.org/manuscript/201812.0345/v1

[8] **M. S. Chang**, "Forensic analysis of Google Drive on Windows," *Int. J. Innov. Sci. Eng. Technol.*, vol. 3, no. 8, pp. 324–331, 2016. [Online]. Available: https://ijiset.com/vol3/v3s8/IJISET_V3_I8_44.pdf

[9] **R. Marty**, "Cloud application logging for forensics," in *Proc. ACM Symp. Appl. Comput. (SAC)*, 2011, pp. 178–184. [Online]. Available: https://pixlcloud.com/applicationlogging.pdf

[10] **R. Shumba**, “Client Forensics: An Assessment of Existing Research and Future Directions,” *Digital Investigation*, vol. 30, 2019, doi: 10.1016/j.diin.2019.04.005

[11] **D. Quick and K.-K. R. Choo**, “Dropbox analysis: Data remnants on user machines,” *Digit. Investig.*, vol. 10, no. 1, pp. 3–18, 2013, doi: 10.1016/j.diin.2013.06.001

[12] **E. Ramadhani and S. I. Isnaindar**, “Digital forensics in Google Drive: Techniques for extracting and analyzing digital artifacts,” *Int. J. Saf. Secur. Eng.*, vol. 14, no. 4, pp. 1203–1211, 2024. [Online]. Available: https://www.iieta.org/download/file/fid/141387

[13] **A. A. Ahmed and C. X. Li**, “Retrieving and Identifying Remnants of Artefacts on Local Storage After Google Drive Cloud Activities on Windows 10 forensics enhancement,” *Sensors*, vol. 25, no. 1, p. 259, Jan. 2024, doi: 10.3390/s25010259

[14] **S. Coulson**, “Investigating Google Drive,” *Forensafe Blog*, Aug. 6, 2021. [Online]. Available: https://forensafe.com

[15] **B. Martini and K.-K. R. Choo**, “Cloud storage forensics: OwnCloud as a case study,” *Digital Investigation*, vol. 10, no. 4, pp. 287–299, Dec. 2013, doi: 10.1016/j.diin.2013.09.003

[16] Sleuth Kit, “Autopsy Python Module Development Guide.” [Online]. Available: https://sleuthkit.org/autopsy/docs/api-docs/4.19/

[17] **A. R. Hevner, S. T. March, J. Park, and S. Ram**, “Design science in information systems research,” *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, Mar. 2004, doi: 10.2307/25148625

[18] **K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee**, “A design science research methodology for information systems research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, Winter 2007–08, doi: 10.2753/MIS0742-1222240302

Appendices

Appendix A – Supplementary screenshots

This appendix provides supplementary screenshots and raw data supporting the implementation and validation discussed in Chapter 4. To ensure clarity and adhere to page limits, only the most essential figures are included in the main dissertation. The figures here display additional steps in environment setup, detailed Autopsy case configuration, extended file operation evidence, and raw database and log outputs.

These materials serve three key purposes:

1. Transparency – Showing that all implementation steps were carried out in a controlled setting, with no intermediate stages skipped.

2. Validation – Offering raw database and log outputs that support the automated parsing demonstrated in the plugin results.

3. Reproducibility – Enabling future researchers to replicate the process by following detailed evidence trails.

References to these figures are made in Chapter 4 and Chapter 5 (e.g., “see Appendix A”) whenever additional evidence was needed but could not be included in the main text due to space constraints.

A.1 Appendix A figure list:

- **Figure A.1:** Google Drive starting screen (initial sync window).
- **Figure A.2:** AppData folder showing DriveFS structure.
- **Figure A.3:** Restored file demonstrating persistence evidence.
- **Figure A.4:** Numbered DriveFS subfolder linked to account ID.
- **Figure A.5:** Logs folder view inside DriveFS directory.
- **Figure A.6:** Alternate log folder structure for DriveFS.
- **Figure A.7:** Autopsy case information screen.
- **Figure A.8:** Host name configuration during case setup in Autopsy.
- **Figure A.9:** Source type selection for Autopsy case.
- **Figure A.10:** Setting evidence file path in Autopsy.
- **Figure A.11:** Ingest module selection in Autopsy case setup.
- **Figure A.12:** Final case setup summary in Autopsy.
- **Figure A.13:** Viewing Autopsy case logs.
- **Figure A.14:** Autopsy volume details view.
- **Figure A.15:** DriveFS location shown within Autopsy case tree.
- **Figure A.16:** Example Google Drive artifact node in Autopsy.
- **Figure A.17:** Metadata database contents displayed in Autopsy.

- **Figure A.18:** ChunksDB and related subfolders in Autopsy.
- **Figure A.19:** Thumbnails cache files in Autopsy.
- **Figure A.20:** Drive_fs.txt log viewed in Autopsy.
- **Figure A.21:** Parent.txt log entries in Autopsy.
- **Figure A.22:** Logs displayed in Autopsy case viewer.
- **Figure A.23:** Metadata timestamps evidence from SQLite DB.
- **Figure A.24:** Exporting metadata database to SQLite Browser.
- **Figure A.25:** Schema of items table in metadata DB.
- **Figure A.26:** Legacy snapshot.db database view.
- **Figure A.27:** Explore items table entries in SQLite.
- **Figure A.28:** Item_properties table entries in SQLite.
- **Figure A.29:** Version checks recorded in metadata DB.
- **Figure A.30:** Deleted/tombstoned items recorded in DB.
- **Figure A.31:** Stable_parents table entries in metadata DB.
- **Figure A.32:** Cross-table joins in metadata database.
- **Figure A.33:** Items and properties combined query results.
- **Figure A.34:** Timeline evidence reconstructed from DB queries.
- **Figure A.35:** Mirror_metadata database – list of tables.
- **Figure A.36:** Mirror_metadata snapshot evidence.
- **Figure A.37:** Metadata_update database contents.
- **Figure A.38:** Metrics_store database overview.
- **Figure A.39:** Startup_trace.json showing session evidence.
- **Figure A.40:** Metric transport threads in startup_trace.json.
- **Figure A.41:** Structured_log_global file showing sync operations.

A.2 Supporting Screenshots

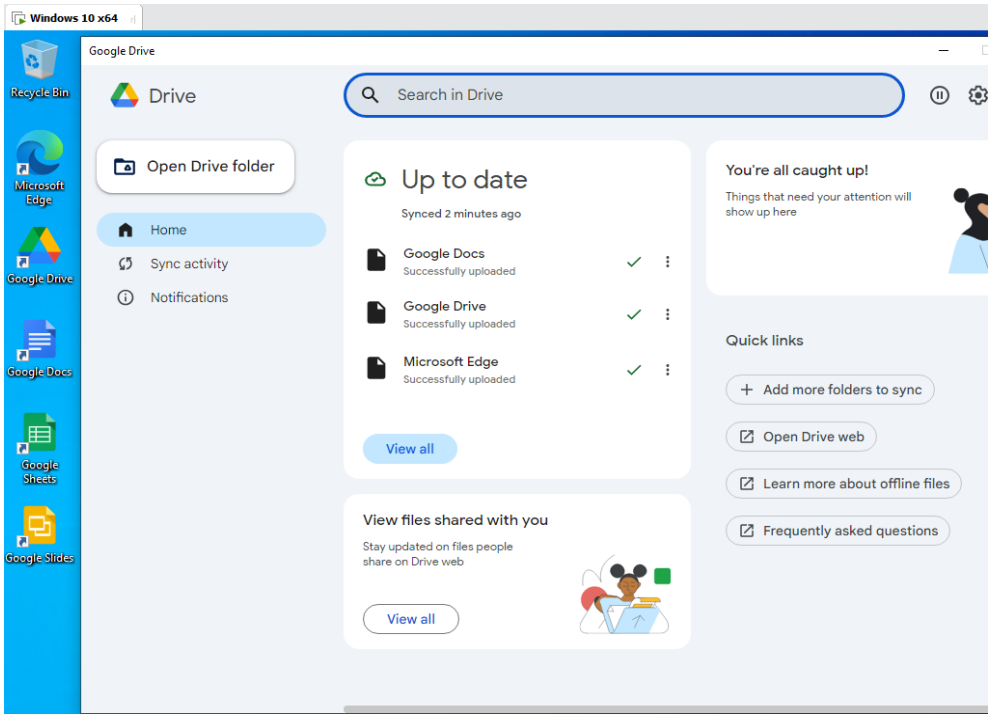


Figure A.1: Google Drive starting screen.

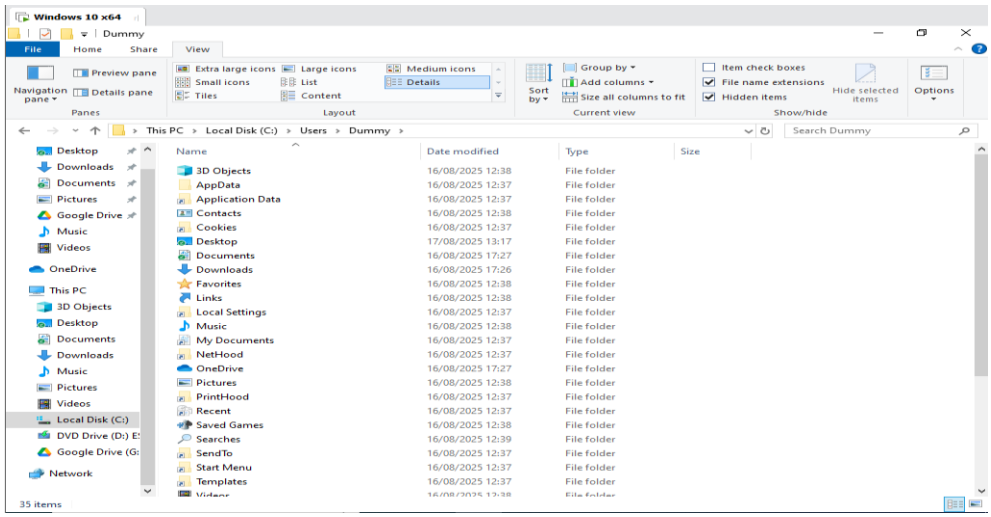


Figure A.2: AppData folder showing DriveFS structure.

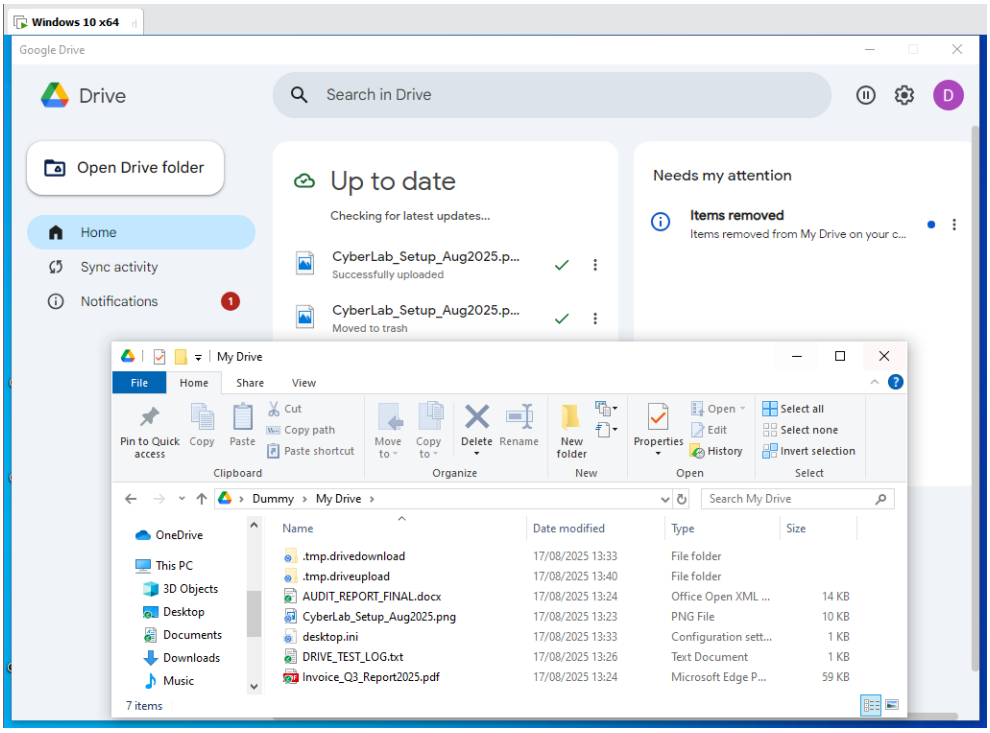


Figure A.3: Restored file demonstrating.

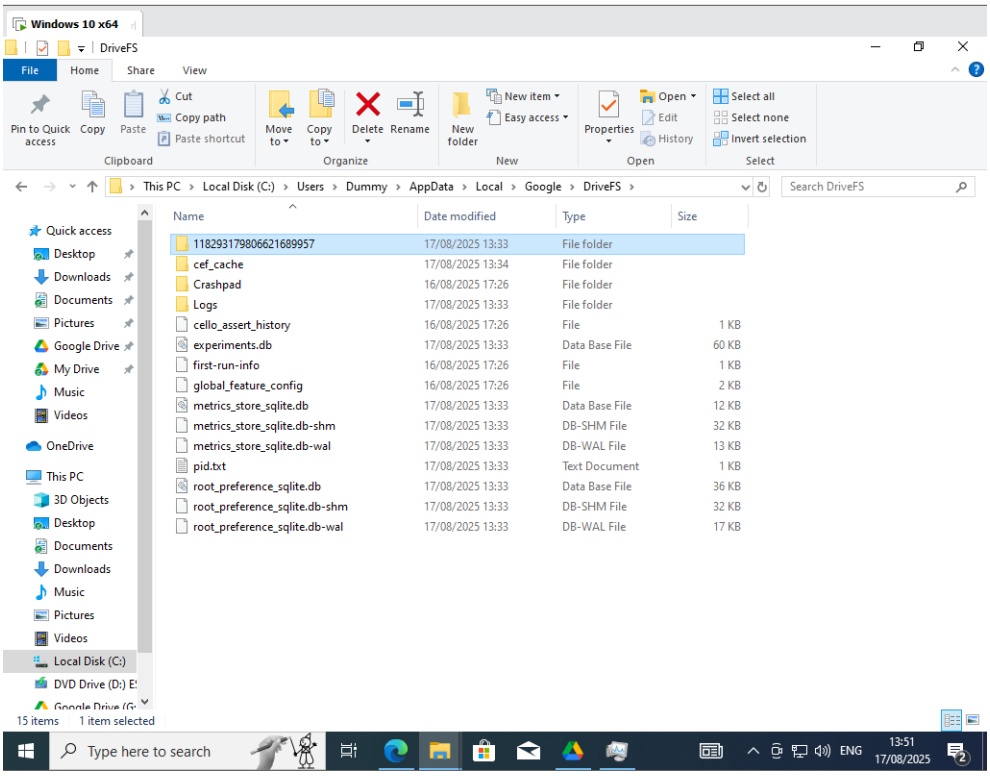


Figure A.4: Numbered DriveFS subfolder ID.

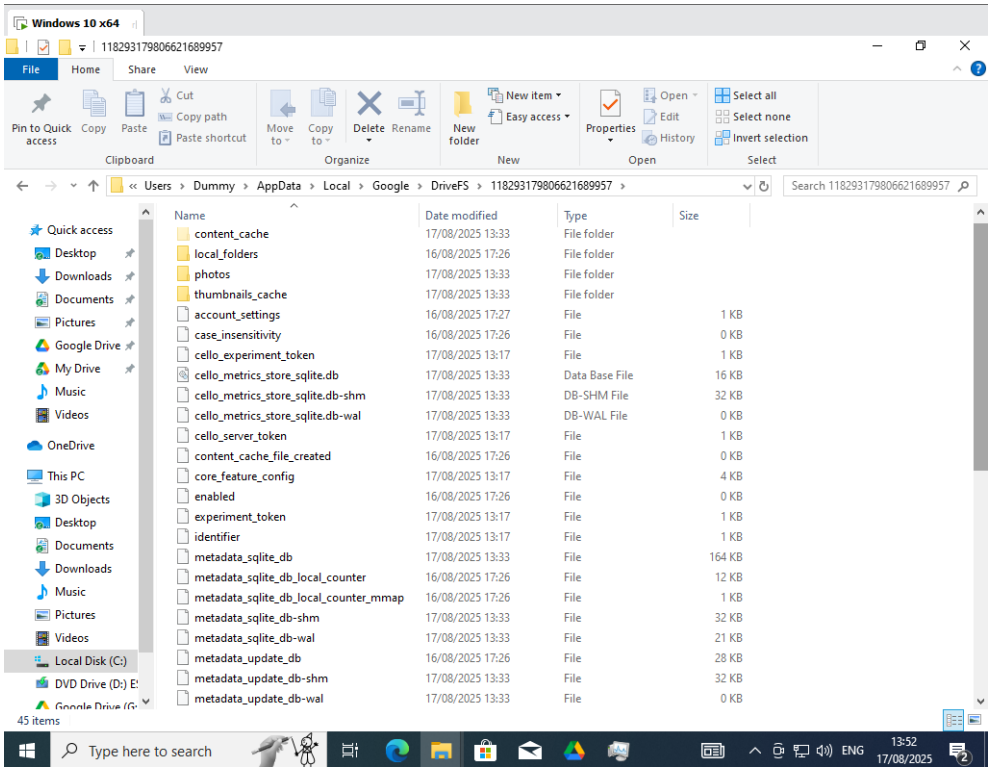


Figure A.5: Logs folder view inside DriveFS directory.

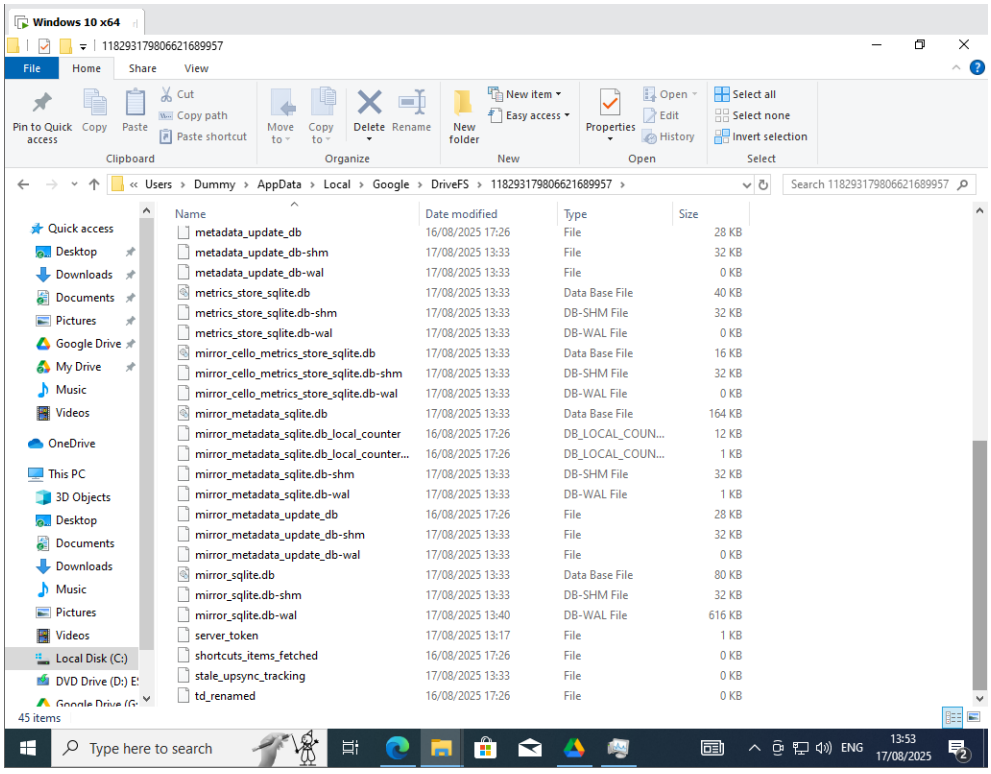


Figure A.6: Alternate log folder structucture for DriveFS.

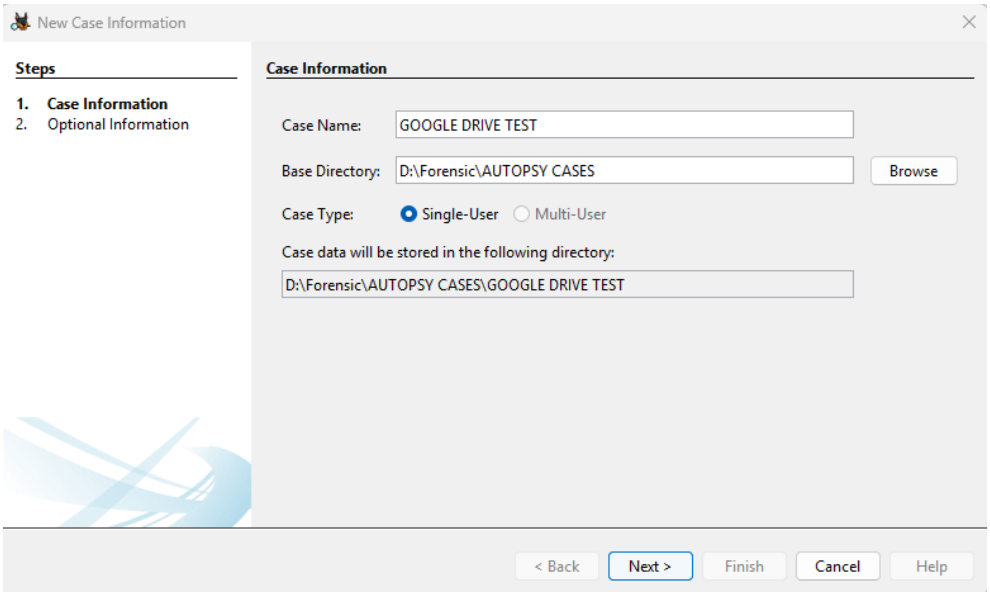


Figure A.7: Autopsy case information screen.

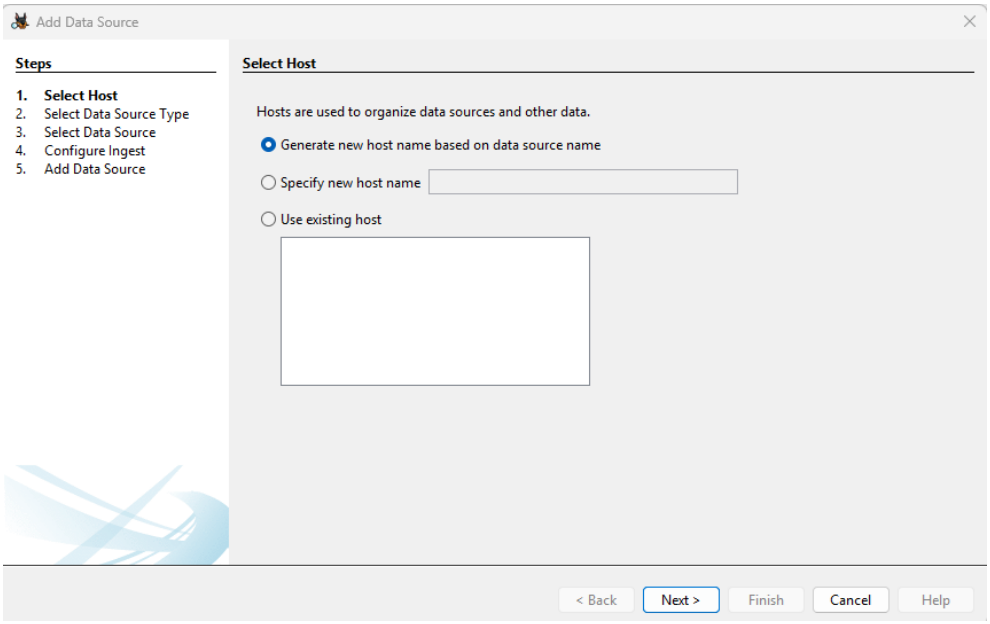


Figure A.8: Host name configuration during case setup in Autopsy.

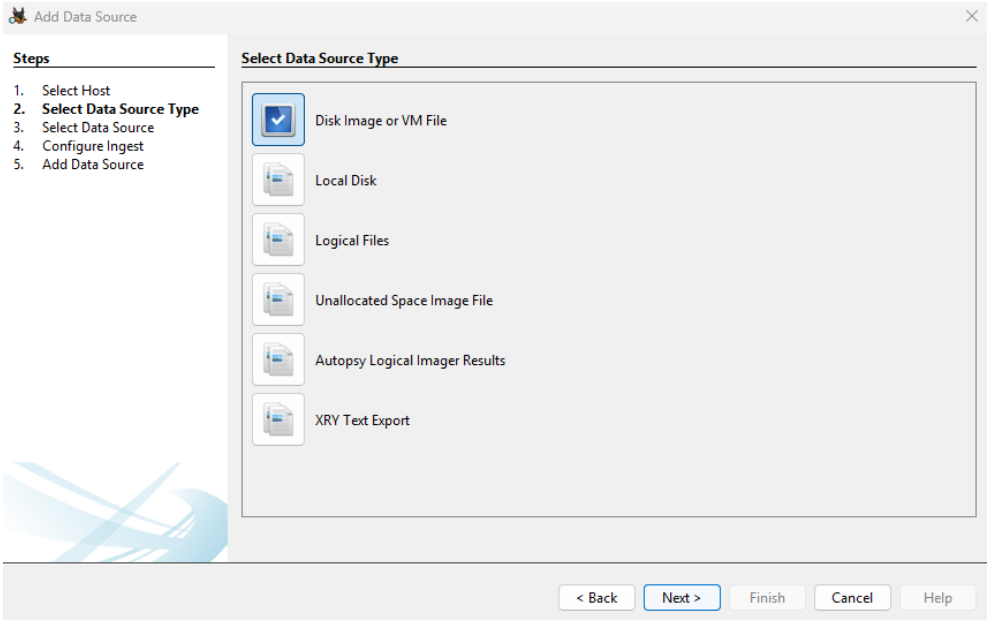


Figure A.9: Source type selection for Autopsy case.

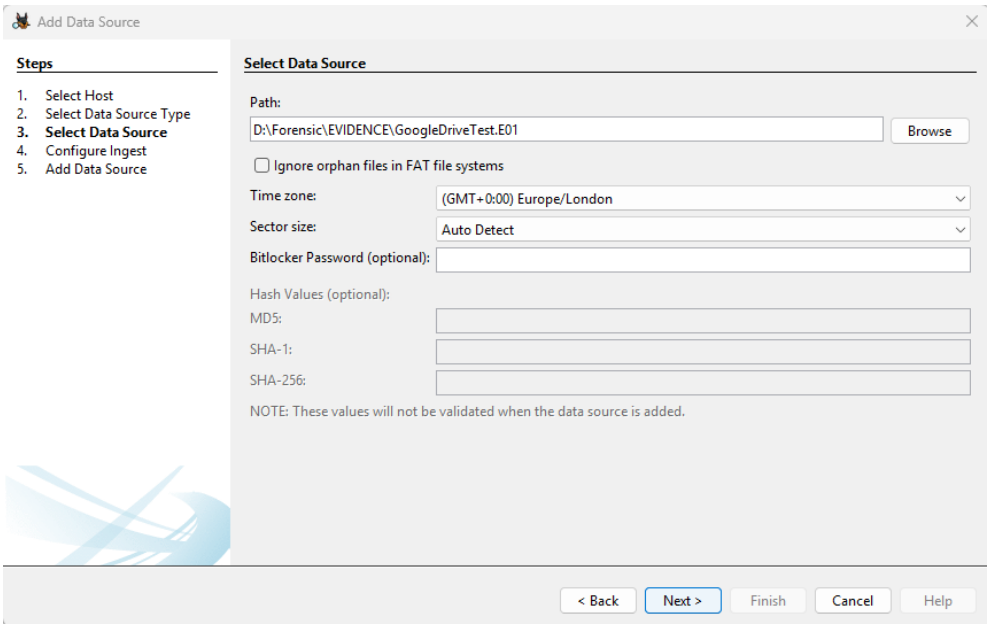


Figure A.10: Setting evidence file path in Autopsy.

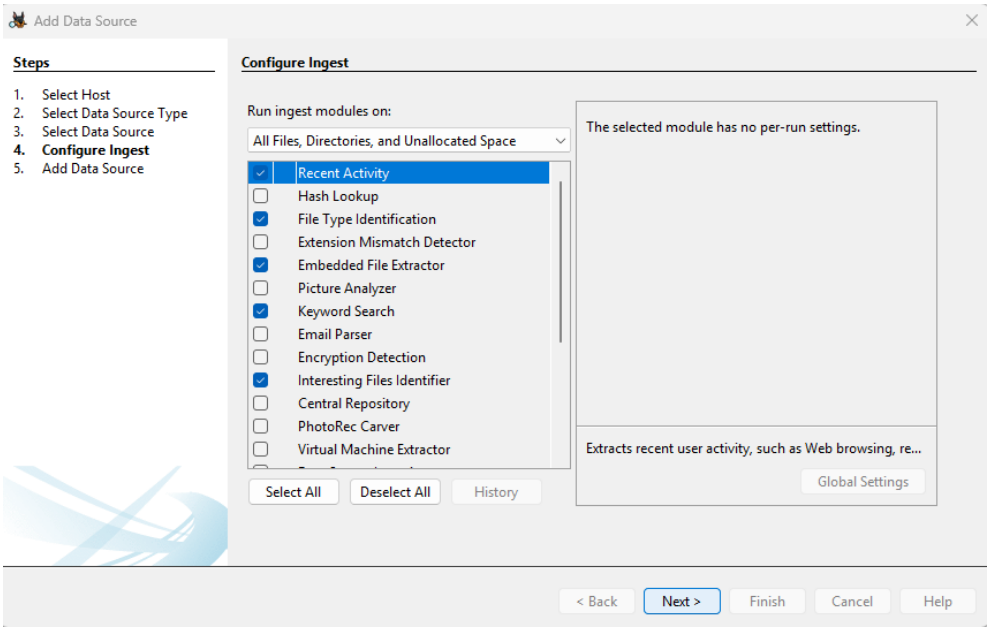


Figure A.11: Ingest module selection in Autopsy case setup.

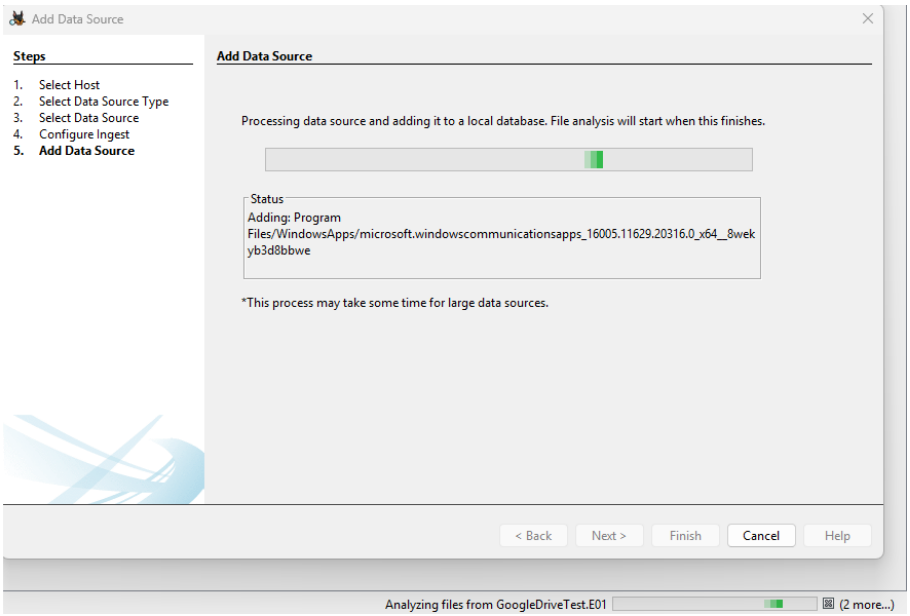


Figure A.12: Final case setup summary in Autopsy.

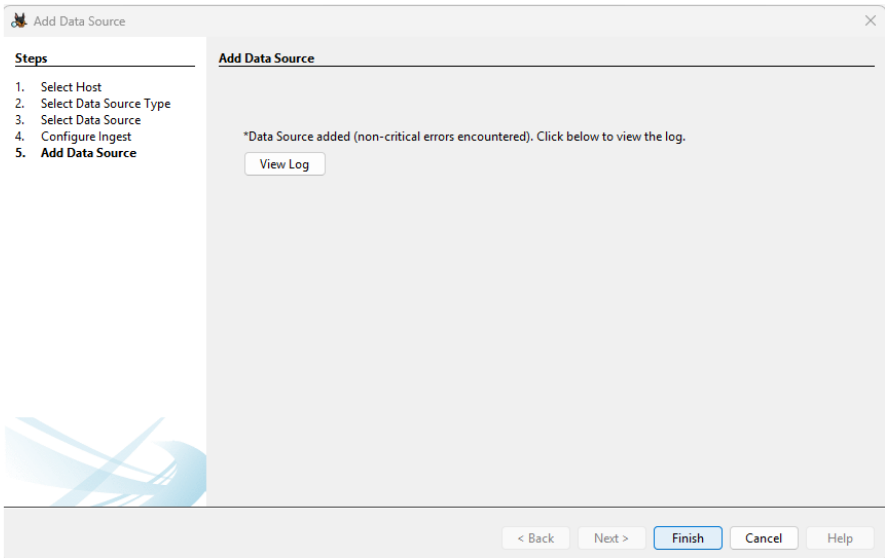


Figure A.13: Viewing Autopsy case logs.

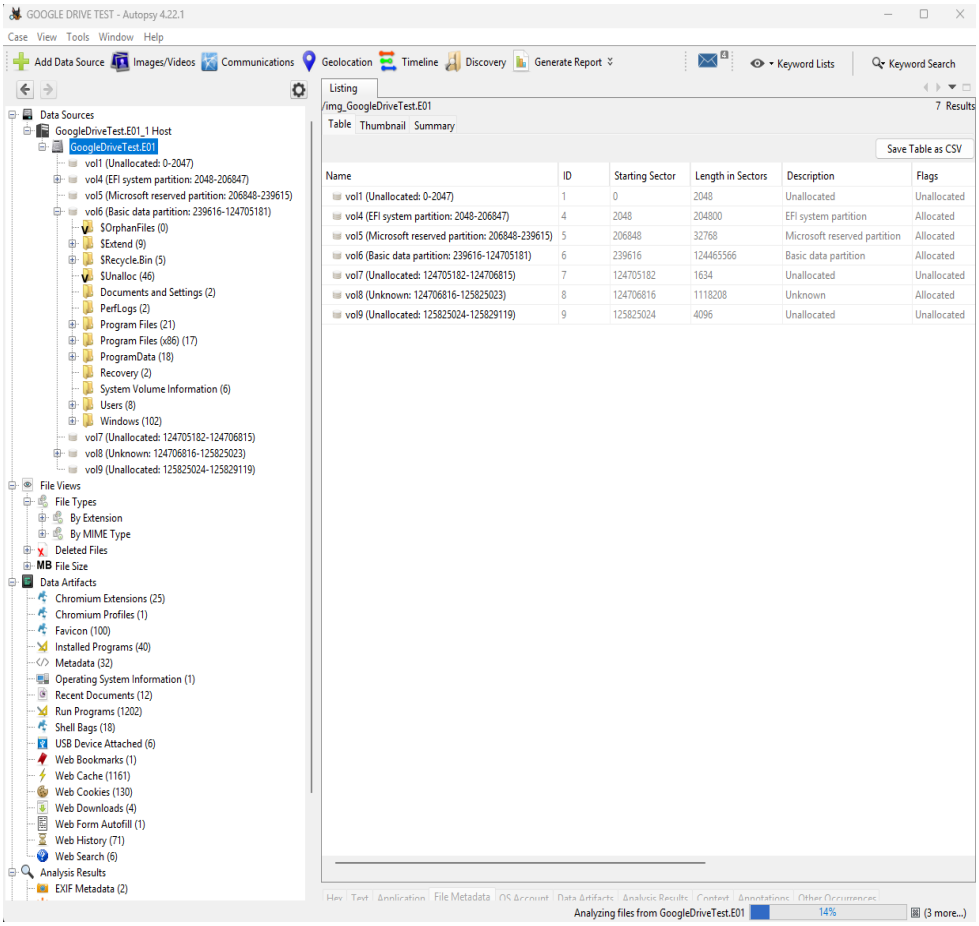


Figure A.14: Autopsy volume details view.

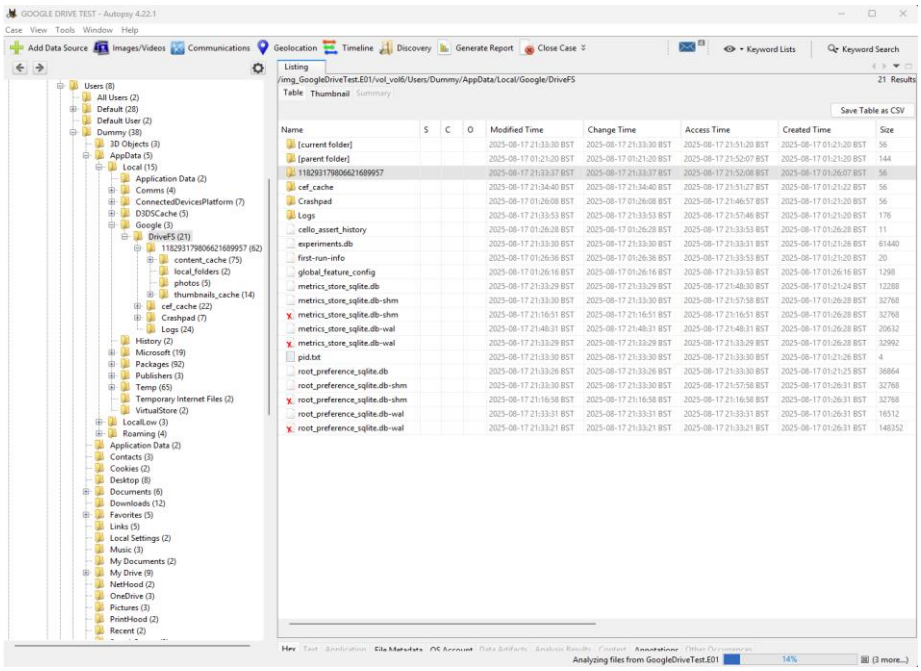


Figure A.15: DriveFS location shown within Autopsy case tree.

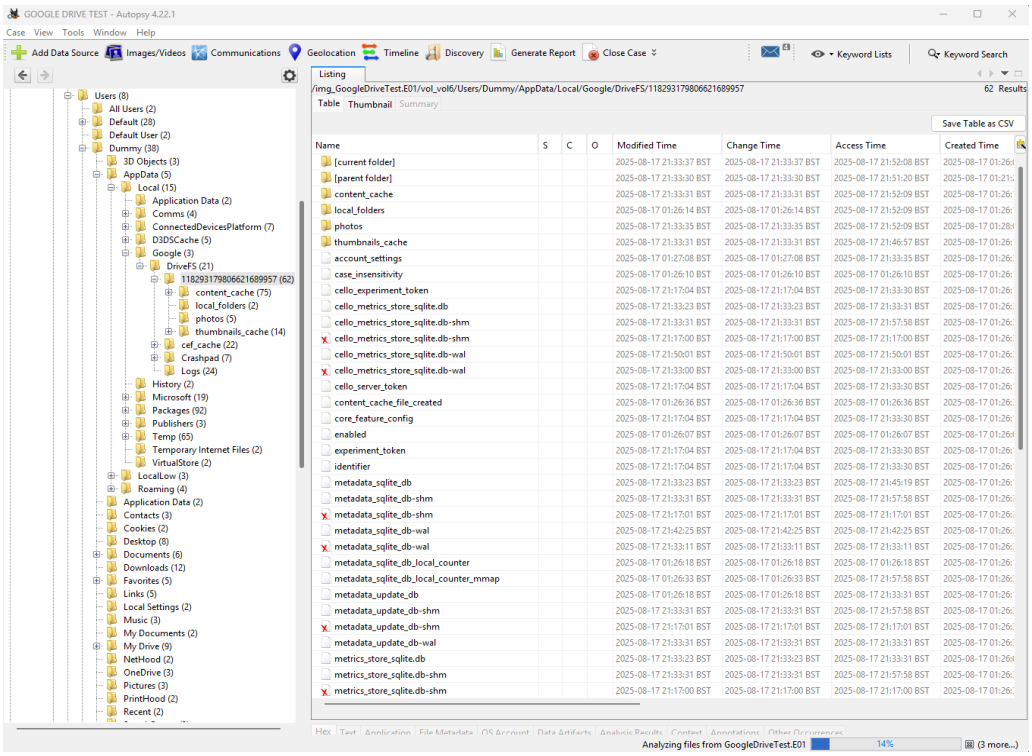


Figure A.16: Example Google Drive artifact node in Autopsy.

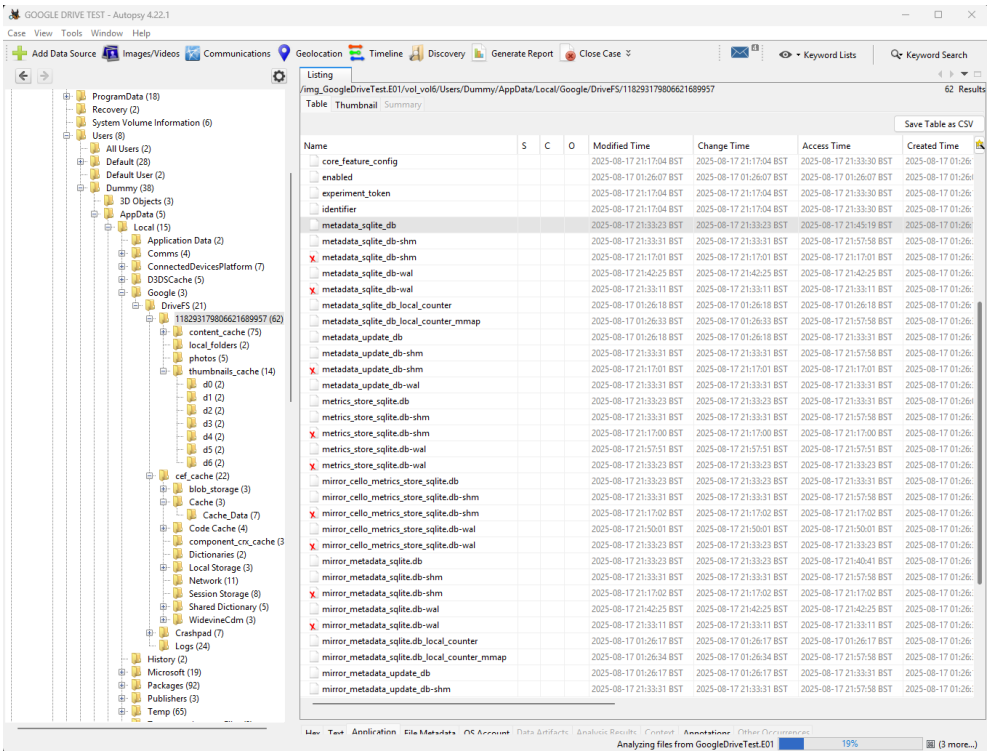


Figure A.17: Metadata database contents displayed in Autopsy.

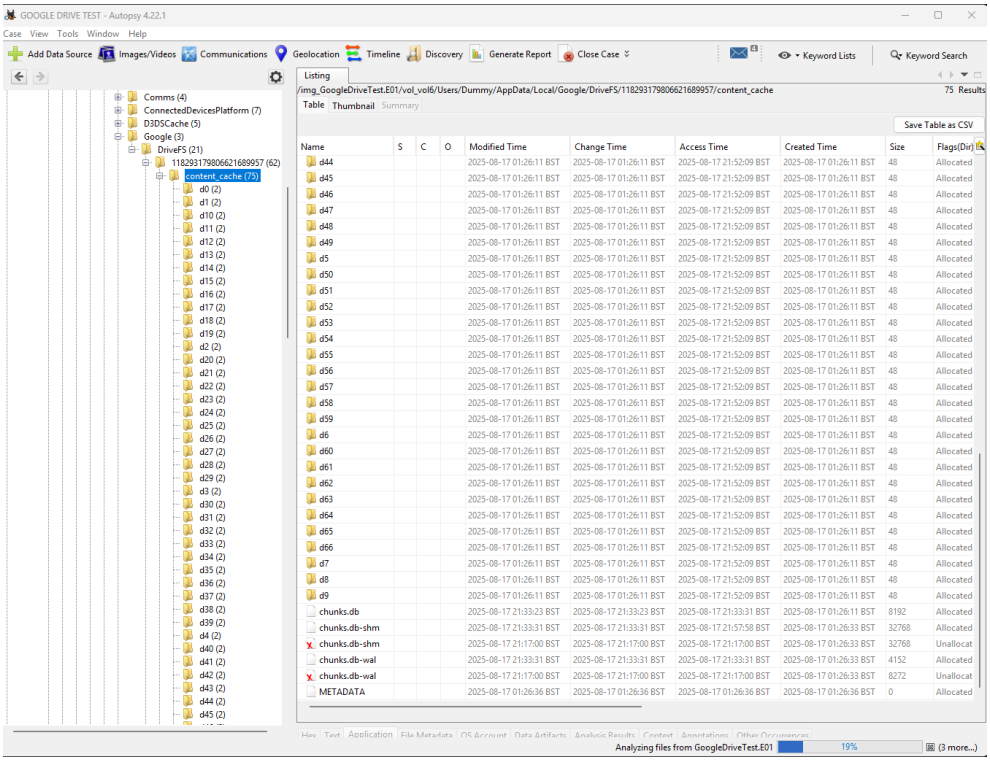


Figure A.18: ChunksDB and related subfolders in Autopsy.

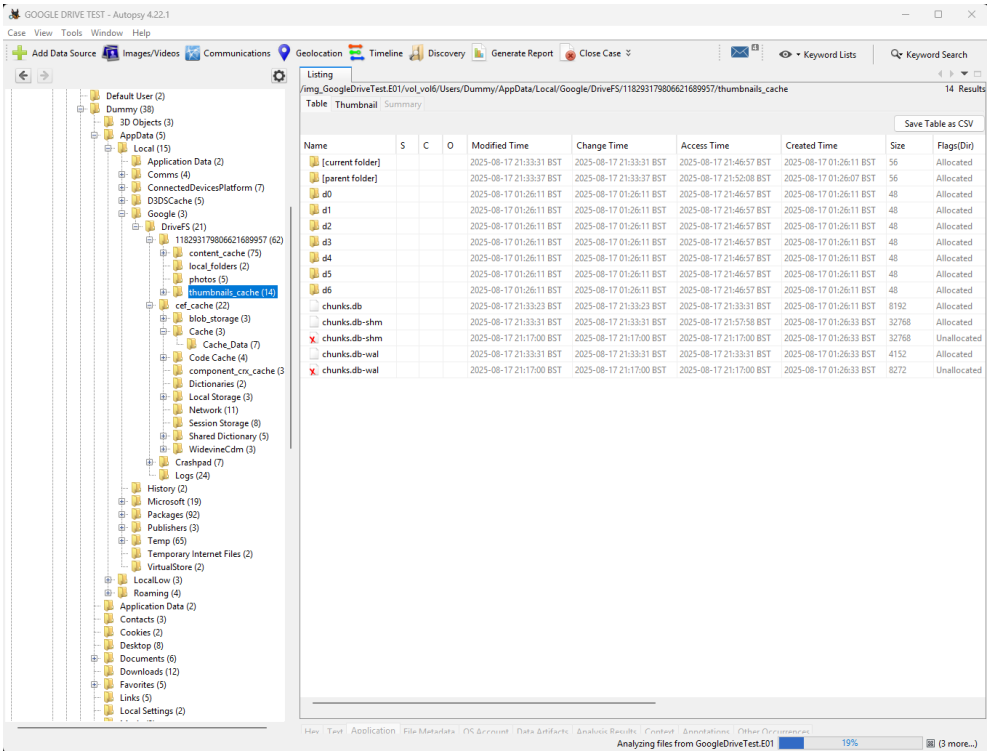


Figure A.19: Thumbnails cache files in Autopsy.

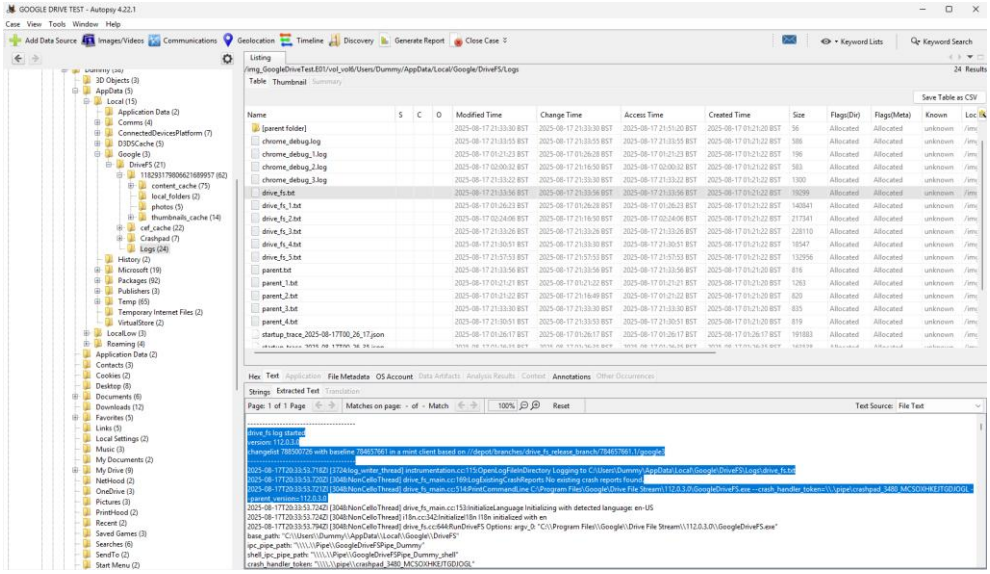


Figure A.20: Drive_fs.txt log viewed in Autopsy.

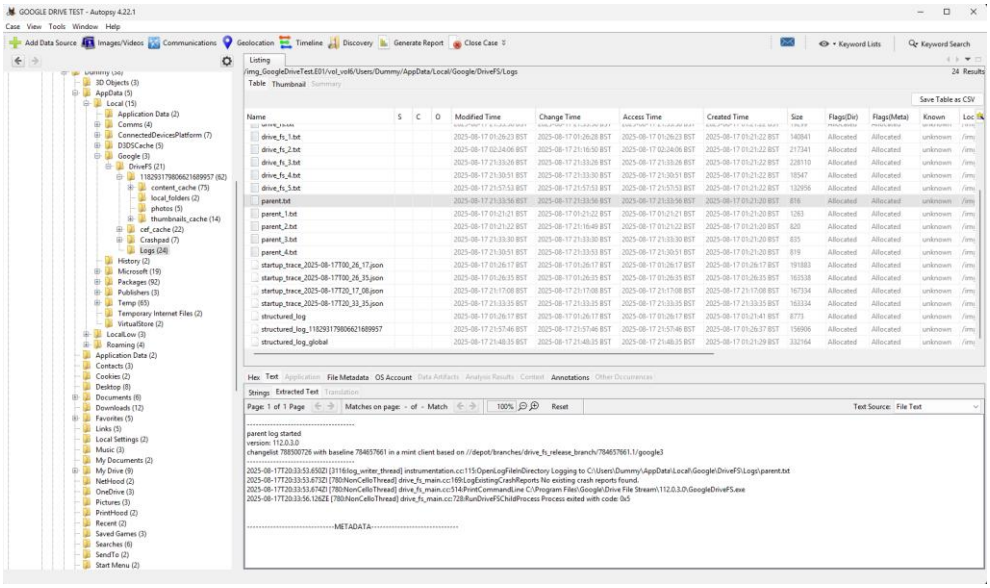


Figure A.21: Parent.txt log entries in Autopsy.

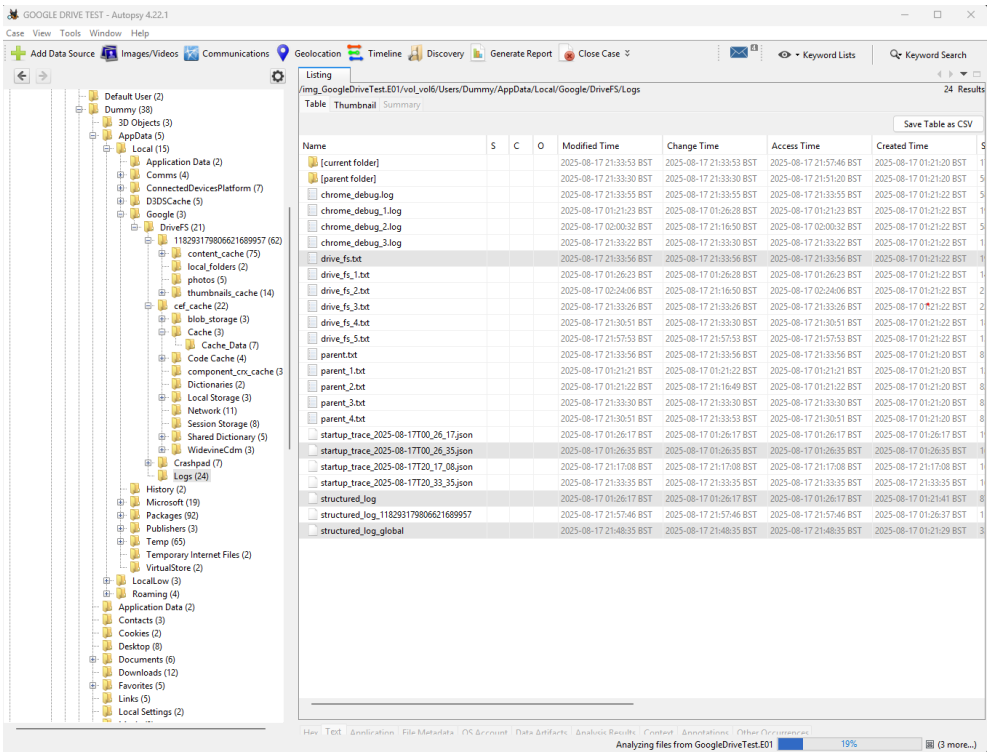


Figure A.22: Logs displayed in Autopsy case viewer.

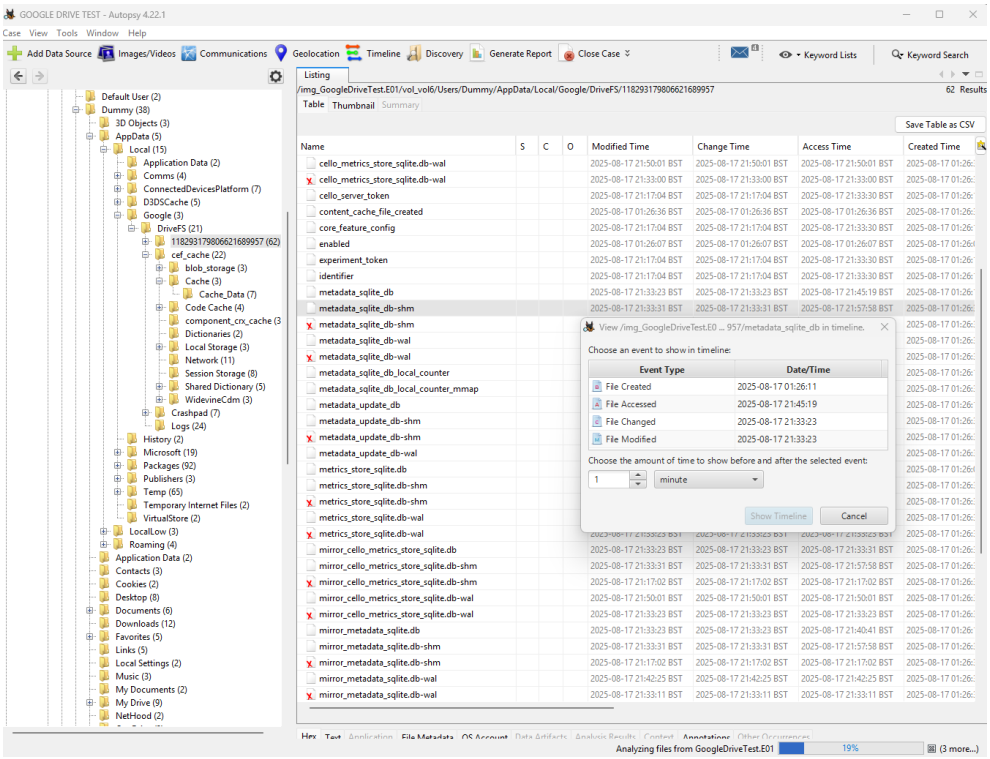


Figure A.23: Metadata timestamps evidence from SQLite DB.

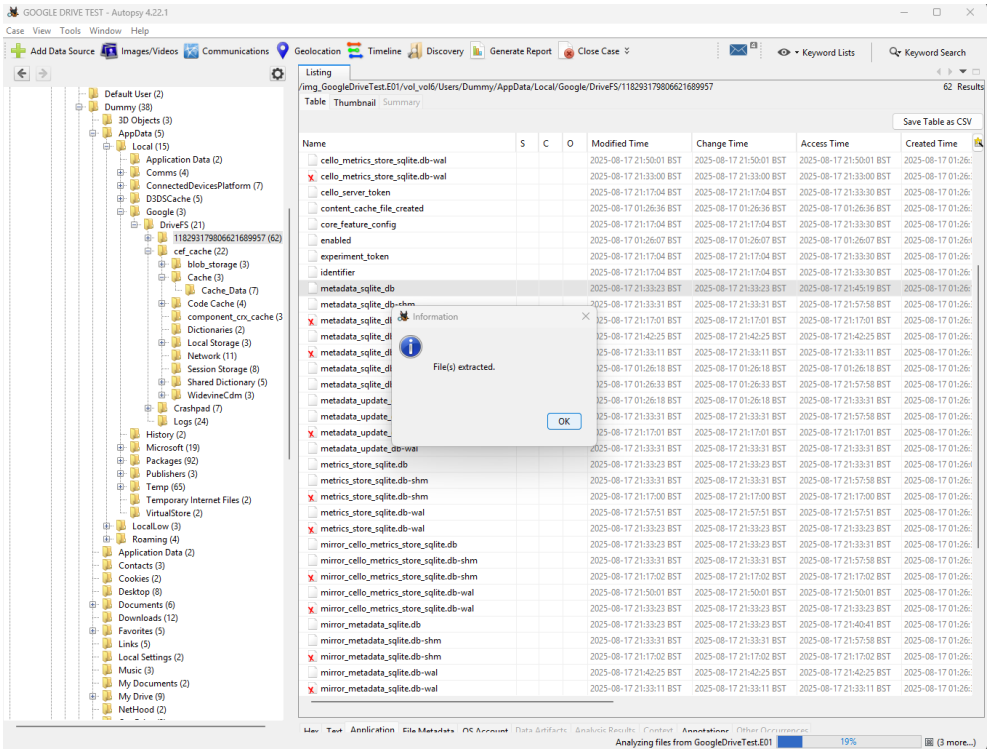


Figure A.24: Exporting metadata database to SQLite Browser.

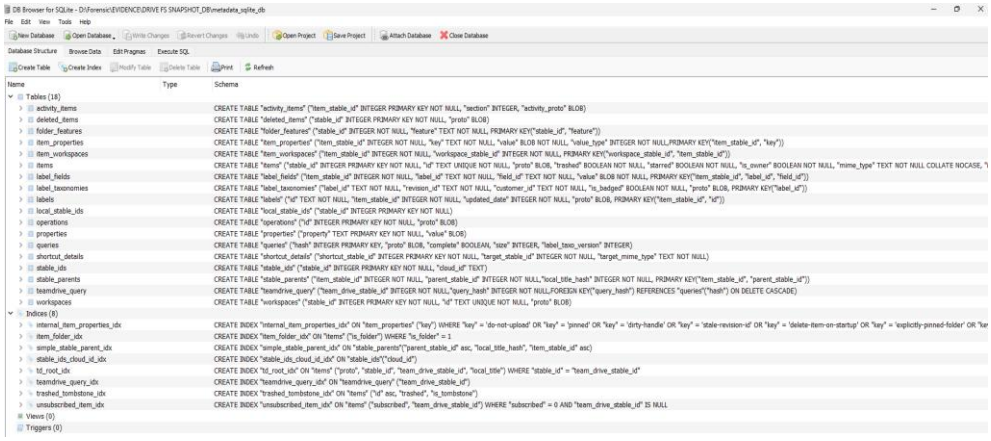


Figure A.25: Schema of items table in metadata DB.

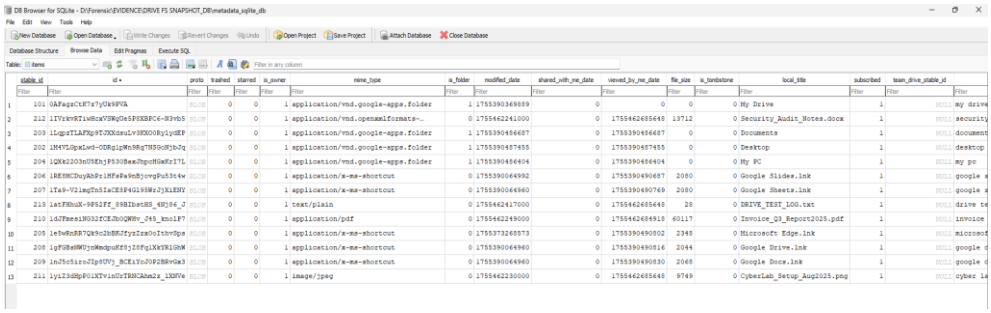
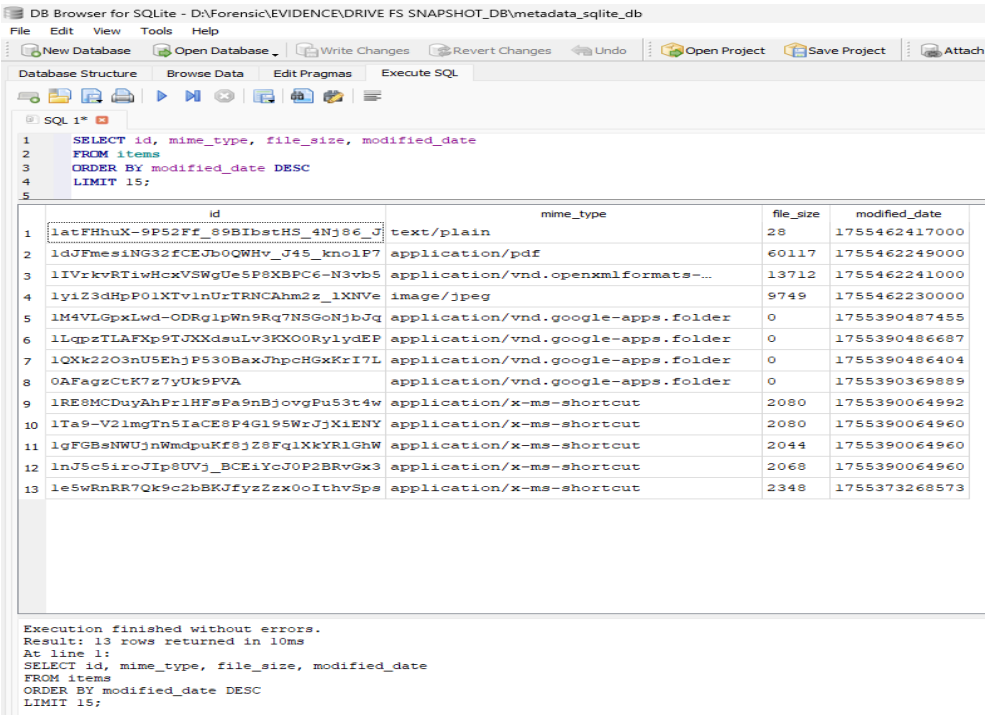


Figure A.26: Legacy snapshot.db database view.



```
SELECT id, mime_type, file_size, modified_date
FROM items
ORDER BY modified_date DESC
LIMIT 15;
```

	id	mime_type	file_size	modified_date
1	latFHhuX-9P52Ff_89BIbstHS_4Nj86_J	text/plain	28	1755462417000
2	ldJFmesiNG32fCEJbOQWHv_J4S_knolP7	application/pdf	60117	1755462249000
3	lIVrkvRTiwHcxVSWgUe5P8XBPC6-N3vb5	application/vnd.openxmlformats-…	13712	1755462241000
4	lyiZ3dHpP0lXTvlnUrTRNCAhm2z_lXNVe	image/jpeg	9749	1755462230000
5	lM4VLGpxLwd-ODRglpWn9Rq7NSGoNjbJq	application/vnd.google-apps.folder	0	1755390487455
6	lLqpzTLAFXp9TJXXdsuLv3KXO0RylydEP	application/vnd.google-apps.folder	0	1755390486687
7	lQXk22O3nU5EhjP530BaxJhpcHGxKrI7L	application/vnd.google-apps.folder	0	1755390486404
8	0AFagzCtK7z7yUk9PVA	application/vnd.google-apps.folder	0	1755390369889
9	lRE8MCDuyAhPrlHFsPa9nBjovgPu53t4w	application/x-ms-shortcut	2080	1755390064992
10	lTa9-V2lmgTn5IaCE8P4Gl95WrJjXiENY	application/x-ms-shortcut	2080	1755390064960
11	lgFGBsNWUjnWmdpuKf8jZ8FqlXkYRlGhW	application/x-ms-shortcut	2044	1755390064960
12	lnJ5c5iroJIp8UVj_BCEiYcJ0P2BRvGx3	application/x-ms-shortcut	2068	1755390064960
13	le5wRnRR7Qk9c2bBKJfyzZzx0oIthvSps	application/x-ms-shortcut	2348	1755373268573

```
Execution finished without errors.
Result: 13 rows returned in 10ms
At line 1:
SELECT id, mime_type, file_size, modified_date
FROM items
ORDER BY modified_date DESC
LIMIT 15;
```

Figure A.27: Explore items table entries in SQLite.

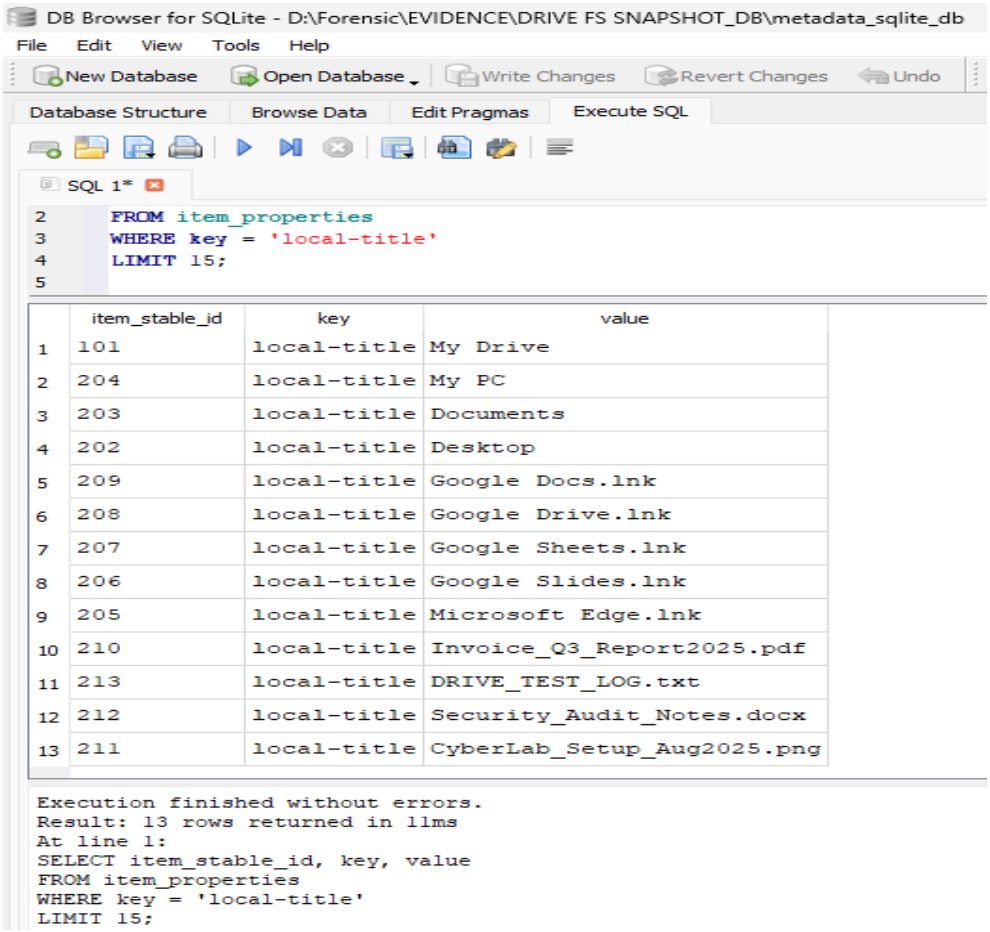


Figure A.28: Item_properties table entries in SQLite.

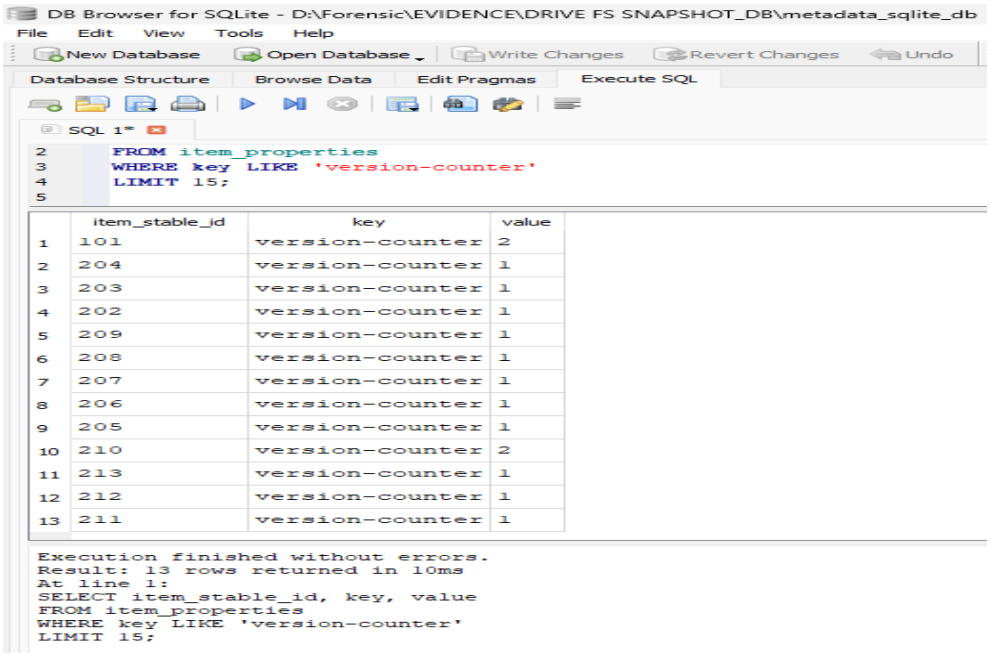


Figure A.29: Version checks recorded in metadata DB.

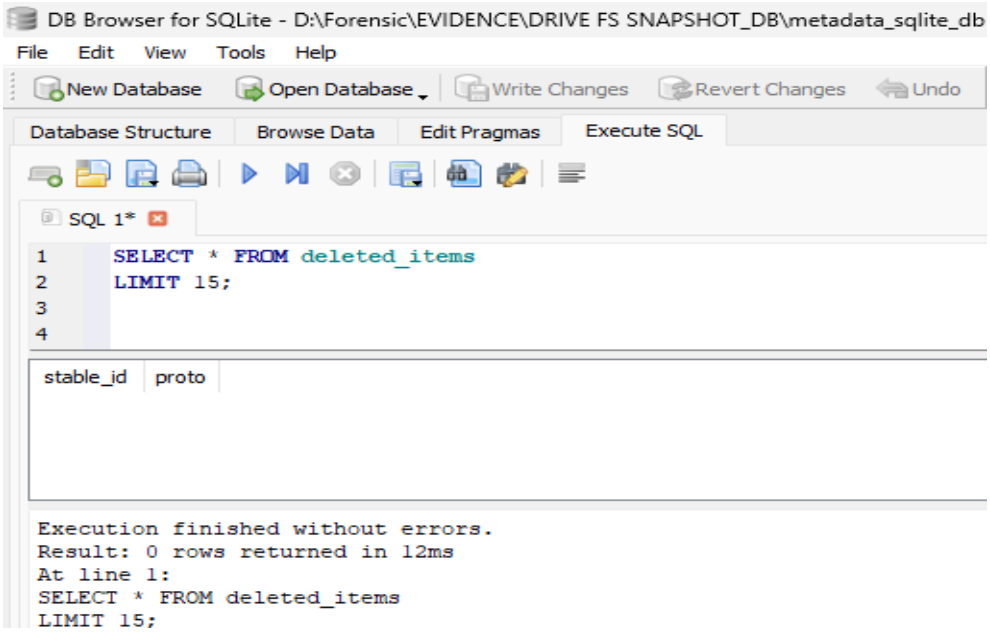


Figure A.30: Deleted/tombstoned items recorded in DB.

DB Browser for SQLite - D:\Forensic\EVIDENCE\DRIVE FS SNAPSHOT_DB\metadata_sqlite_db

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Undo

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: stable_parents

	item_stable_id	parent_stable_id	local_title_hash
	Filter	Filter	Filter
1	203	204	2564741765321385298
2	202	204	-4626577355884615266
3	209	202	-5892688849694393053
4	208	202	-412611428192853809
5	207	202	-3548870981148260148
6	206	202	-5924762511896943503
7	205	202	-8418127194023969835
8	210	101	-3632495335348955792
9	213	101	6969104921994090244
10	212	101	1651005224615080228
11	211	101	-827413738411368811

Figure A.31: Stable_parents table entries in metadata DB.

DB Browser for SQLite - D:\Forensic\EVIDENCE\DRIVE FS SNAPSHOT_DB\metadata_sqlite_db

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Undo | Open Project | Save Project | Attach Database | Close Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

SQL 1*

```
SELECT i.id, p.value AS local_title, i.mime_type, i.file_size, i.modified_date, v.value AS version
FROM items i
JOIN item_properties p ON i.stable_id = p.item_stable_id AND p.key='local-title'
LEFT JOIN item_properties v ON i.stable_id = v.item_stable_id AND v.key='version-counter'
ORDER BY i.modified_date DESC;
```

	id	local_title	mime_type	file_size	modified_date	version
1	1atFHhuX-9P52Ff_89BIbstHS_4Nj86_J	DRIVE_TEST_LOG.txt	text/plain	28	1755462417000	1
2	1dJFmesiNG32fCEJb0QWHv_J45_knolP7	Invoice_Q3_Report2025.pdf	application/pdf	60117	1755462249000	2
3	1IVrkvRTiwHcxVSWgUe5P8XBPC6-N3vb5	Security_Audit_Notes.docx	application/vnd.openxmlformats-…	13712	1755462241000	1
4	1yiZ3dHpP01XTvlnUrTRNCAhm2z_1XNVe	CyberLab_Setup_Aug2025.png	image/jpeg	9749	1755462230000	1
5	1M4VLGpxLwd-ODRglpWn9Rq7NSGoNjbJq	Desktop	application/vnd.google-apps.folder	0	1755390487455	1
6	1LqpzTLAFXp9TJXXdsuLv3KXO0RylydEP	Documents	application/vnd.google-apps.folder	0	1755390486687	1

```
Execution finished without errors.
Result: 13 rows returned in 110ms
At line 1:
SELECT i.id, p.value AS local_title, i.mime_type, i.file_size, i.modified_date, v.value AS version
FROM items i
JOIN item_properties p ON i.stable_id = p.item_stable_id AND p.key='local-title'
LEFT JOIN item_properties v ON i.stable_id = v.item_stable_id AND v.key='version-counter'
ORDER BY i.modified_date DESC;
```

Figure A.32: Cross-table joins in metadata database.

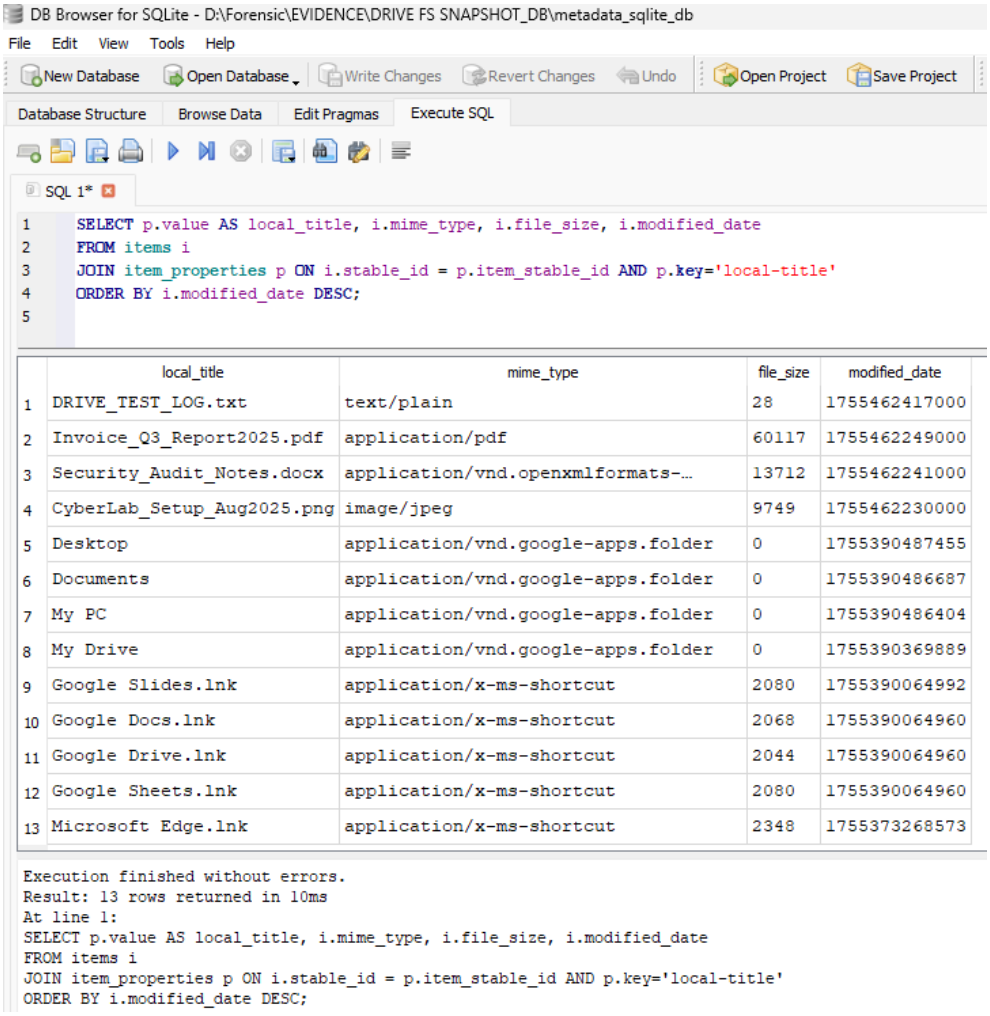


Figure A.33: Items and properties combined query results.

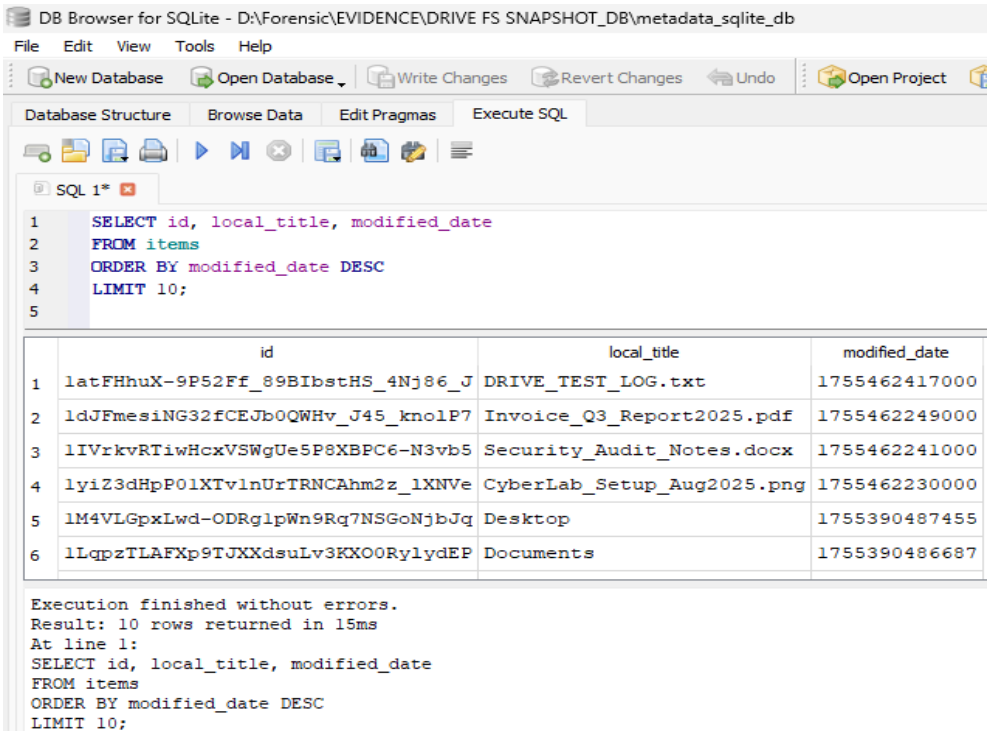


Figure A.34: Timeline evidence reconstructed from DB queries.

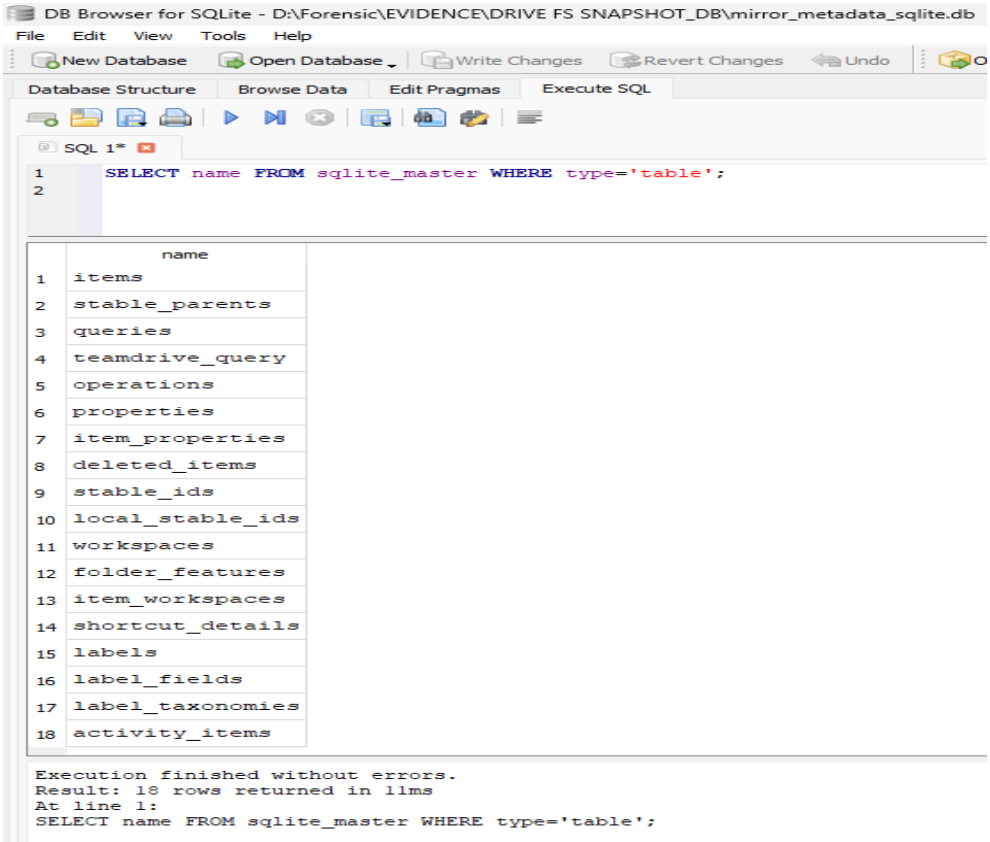


Figure A.35: Mirror_metadata database – list of tables.

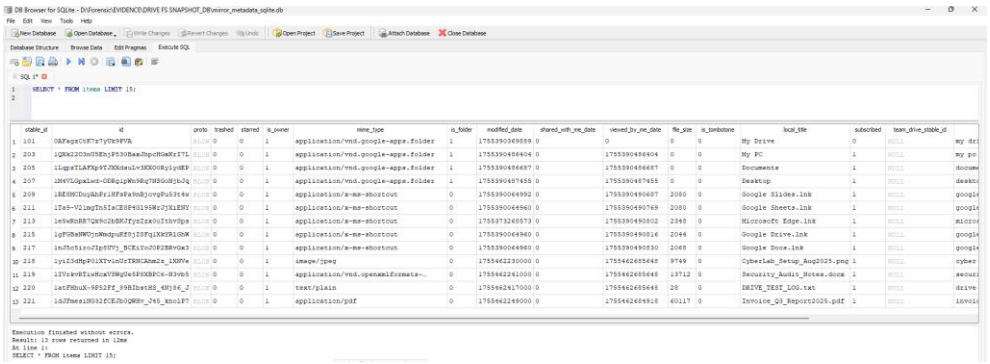


Figure A.36: Mirror_metadata snapshot evidence.

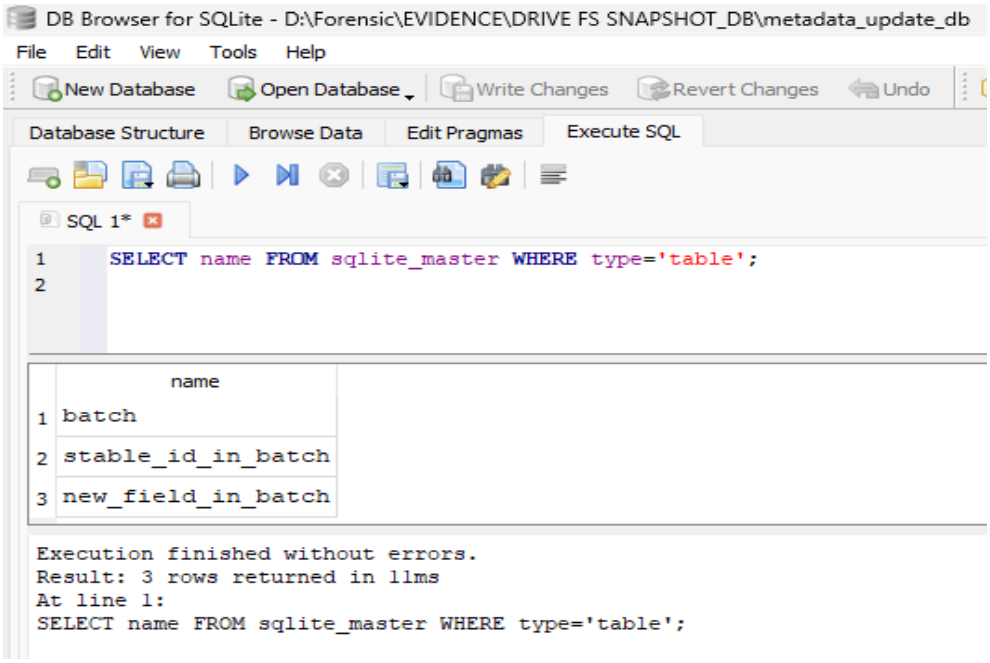


Figure A.37: Metadata_update database contents.

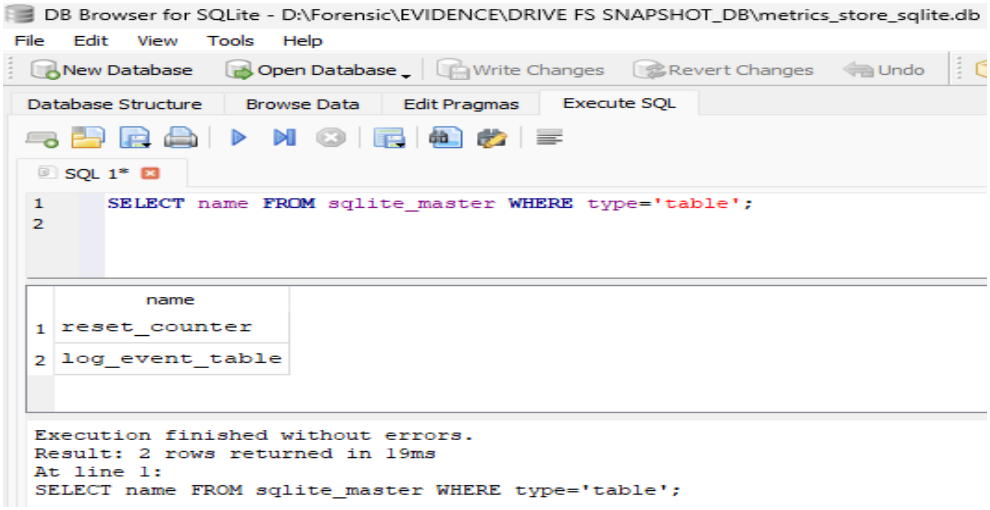


Figure A.38: Metrics_store database overview.

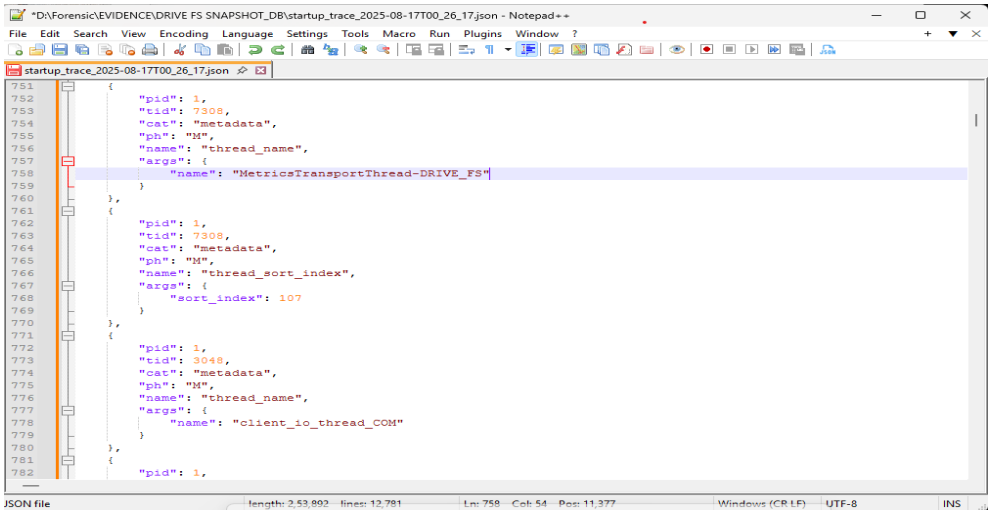


Figure A.39: Startup_trace.json showing session evidence.

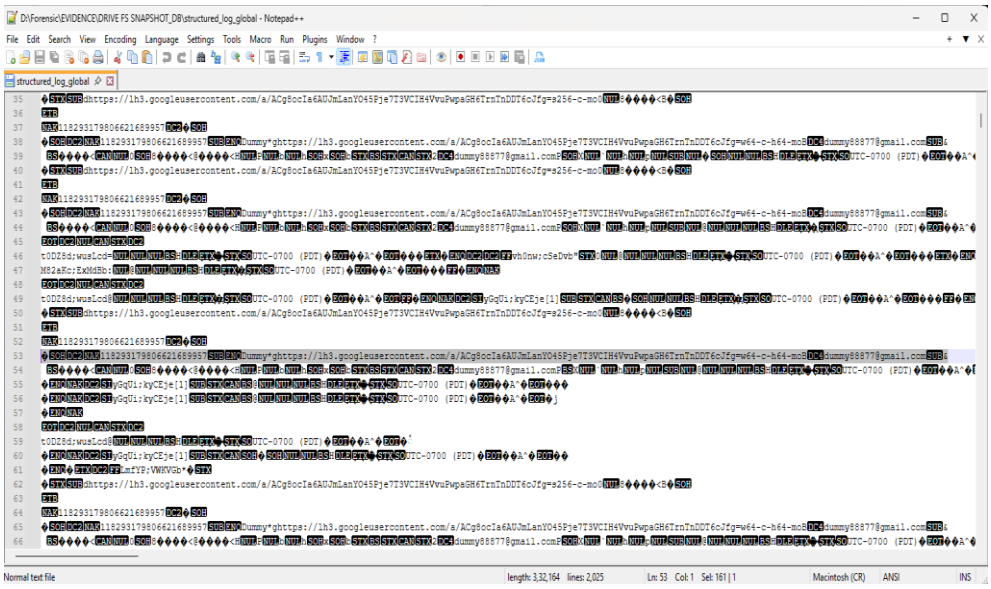


Figure A.40: Metric transport threads in startup_trace.json.

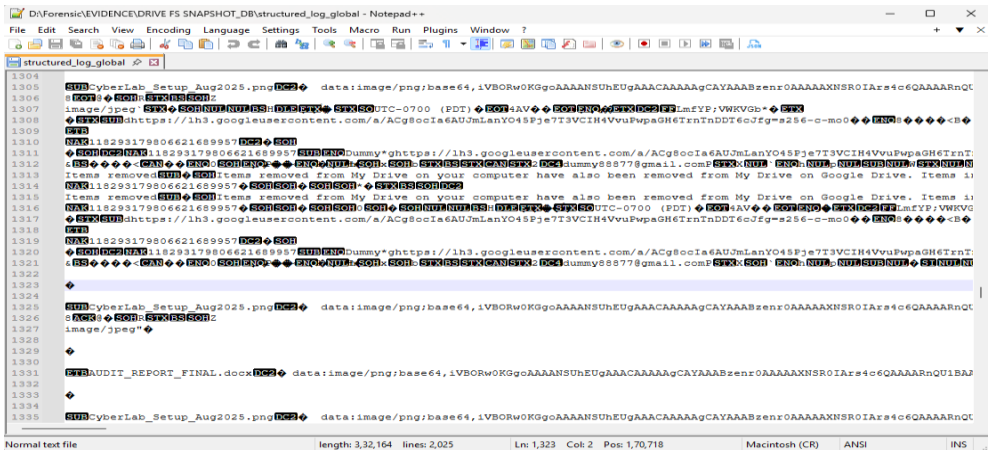


Figure A.41: Structured_log_global file showing sync operations.

A.3 Reproducibility Checklist

The following protocol was used to reproduce the results in this project:

1. Configure a Windows 10 VM (4 GB RAM, 60 GB Disk)
2. Install Google Drive for Desktop, FTK Imager, Autopsy 4.22 with Python Support, SQLite DB Browser.
3. Performed Controlled file actions such as create, rename, delete and restore.
4. Capture VM image using FTK Imager and then ingest into Autopsy.
5. Deploy Parser (.py) in Autopsy's python_modules folder.
6. Run the ingest process and export results.
7. Cross-validate the parser outputs with manual SQL queries.

Appendix B – Google DriveFS Autopsy Plugin Source Code

This appendix contains the complete source code of the custom Autopsy ingest module designed for parsing Google Drive for Desktop (DriveFS) artifacts. Written in Python (compatible with Jython), it integrates seamlessly with Autopsy's ingest framework. The source code is included to promote reproducibility and enable other researchers to modify or build upon the plugin. Testing was conducted using only dummy datasets to adhere to ethical and legal standards.

B.1 Module File Location

The plugin .py file was placed in:
C:\Users\james\AppData\Roaming\autopsy\python_modules\GoogleDriveIngestModule

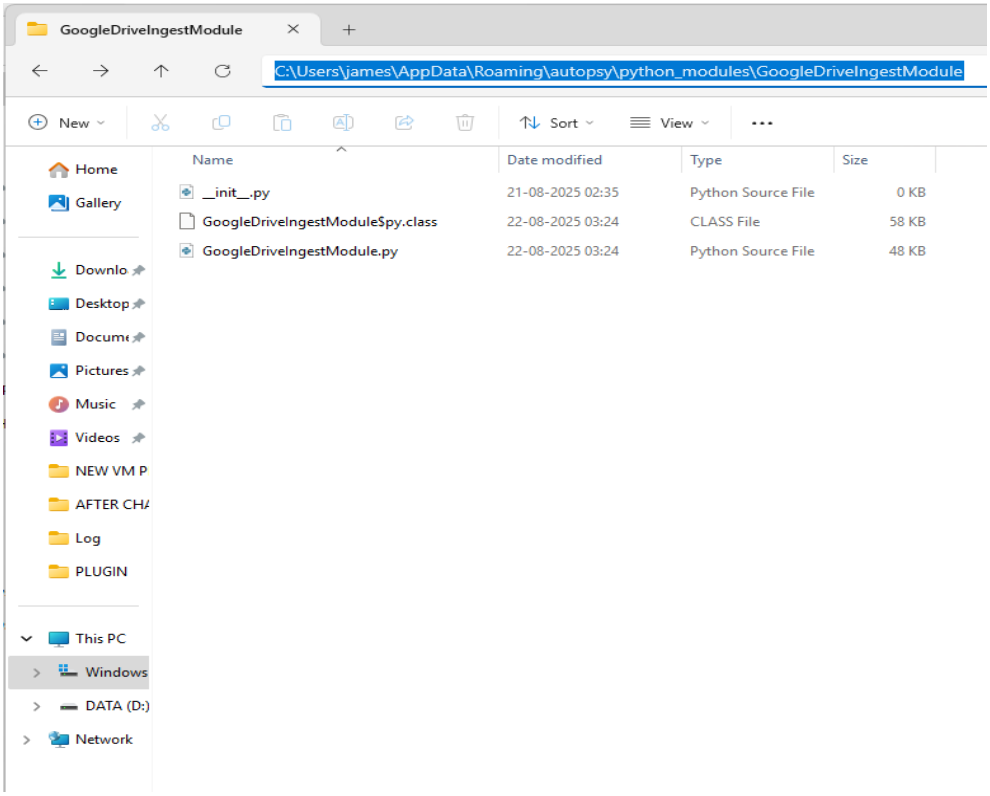


Figure B.1: Custom Google Drive ingest module placed in Autopsy's Python modules directory.

File List

- __init__.py — Python initialiser (empty).
- GoogleDriveIngestModule.py — Main parser implementation (~48 KB).

- GoogleDriveIngestModule$py.class — Auto-generated Jython class (compiled by Autopsy).

B.2 Source Code (GoogleDriveIngestModule.py)

The project submission includes the source code for the developed Autopsy ingest module (digital copy).
Below is an example excerpt showing the initialisation, factory setup, and parser structure.
This illustrates how the ingest module loads artifacts, integrates with Autopsy's pipeline, and manages errors.

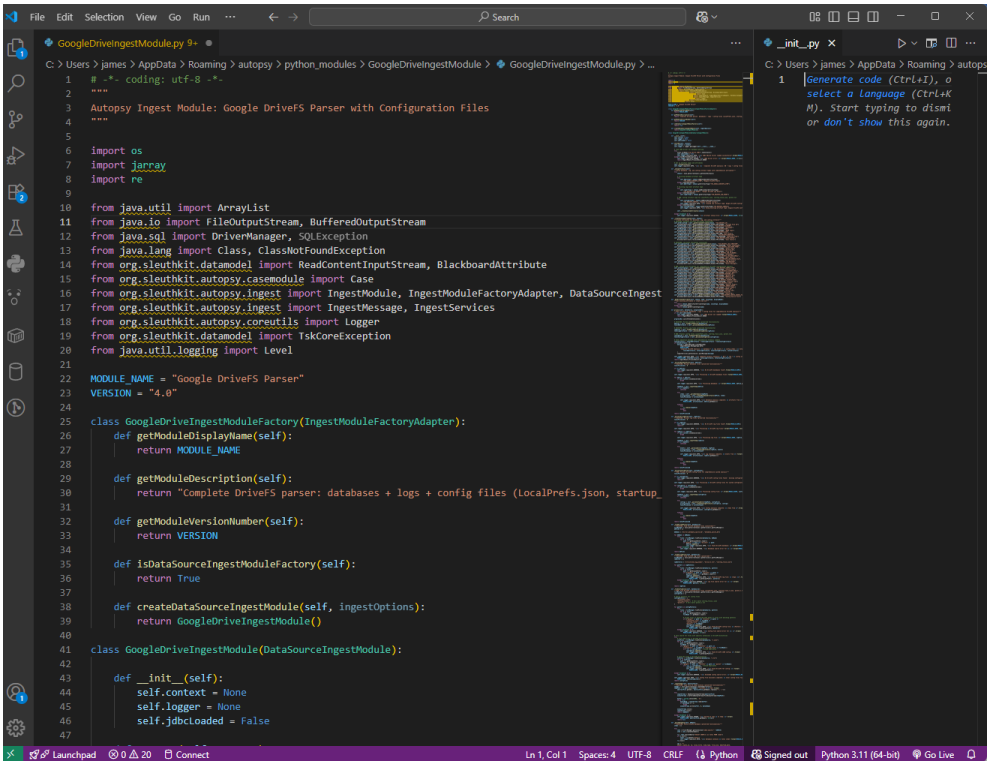


Figure B.2: Source Code.

Due to space limitations, only a representative sample of the code is shown here.
The complete source code, including parsing routines for SQLite, JSON, and log files, is available in the submitted digital archive.