Challenge 4 – Bakery's Secret Recipe
Git Forensics Report

1. Problem Understanding

The task was to analyze a provided folder belonging to an internal bakery project and recover a hidden flag in the format `SAIC{...}`.

The objective was to perform forensic analysis on the folder and identify any hidden or deleted sensitive information.

2. Initial Observation

After extracting the provided folder, I inspected its contents to understand the structure of the given data.

```
ls -Force
```

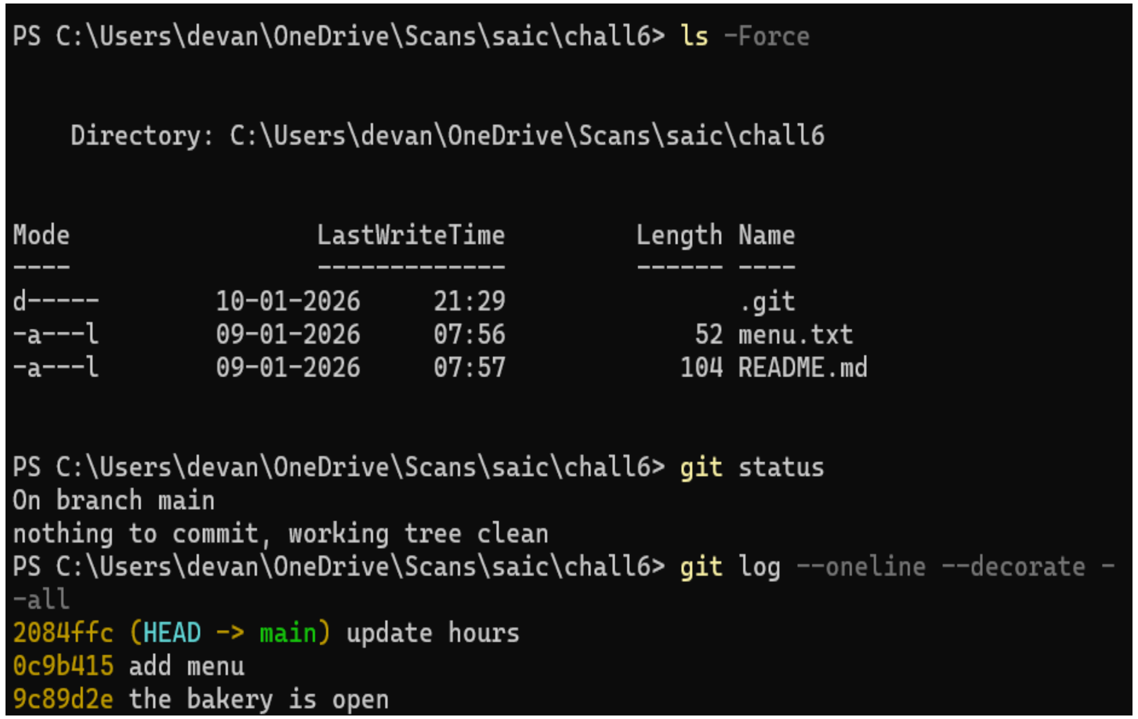
The output showed three main items:

- `.git`
- `README.md`
- `menu.txt`

The presence of the `.git` directory confirmed that the folder was a **Git repository**, as mentioned in the problem statement.

Since Git repositories often retain historical and deleted data, I decided to analyze the repository using Git forensic techniques instead of only checking visible files.

(At this stage, it was clear that simple file reading would not be sufficient, and deeper repository inspection was required.)



```
PS C:\Users\devan\OneDrive\Scans\saic\chall6> ls -Force


    Directory: C:\Users\devan\OneDrive\Scans\saic\chall6


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10-01-2026     21:29                .git
-a---l        09-01-2026     07:56             52 menu.txt
-a---l        09-01-2026     07:57            104 README.md


PS C:\Users\devan\OneDrive\Scans\saic\chall6> git status
On branch main
nothing to commit, working tree clean
PS C:\Users\devan\OneDrive\Scans\saic\chall6> git log --oneline --decorate -
-all
2084ffc (HEAD -> main) update hours
0c9b415 add menu
9c89d2e the bakery is open
```

3. Repository State Check

To verify the current repository state, I ran:

`git status`

Output:

`working tree clean`

This confirmed that no uncommitted or modified files existed in the current snapshot.

Hence, the flag was not directly present in the visible working files.

4. Commit History Analysis

I extracted the commit history:

```
git log --oneline --decorate --all
```

Commits found:

```
9c89d2e – the bakery is open
0c9b415 – add menu
2084ffc – update hours
```

Each commit was inspected individually:

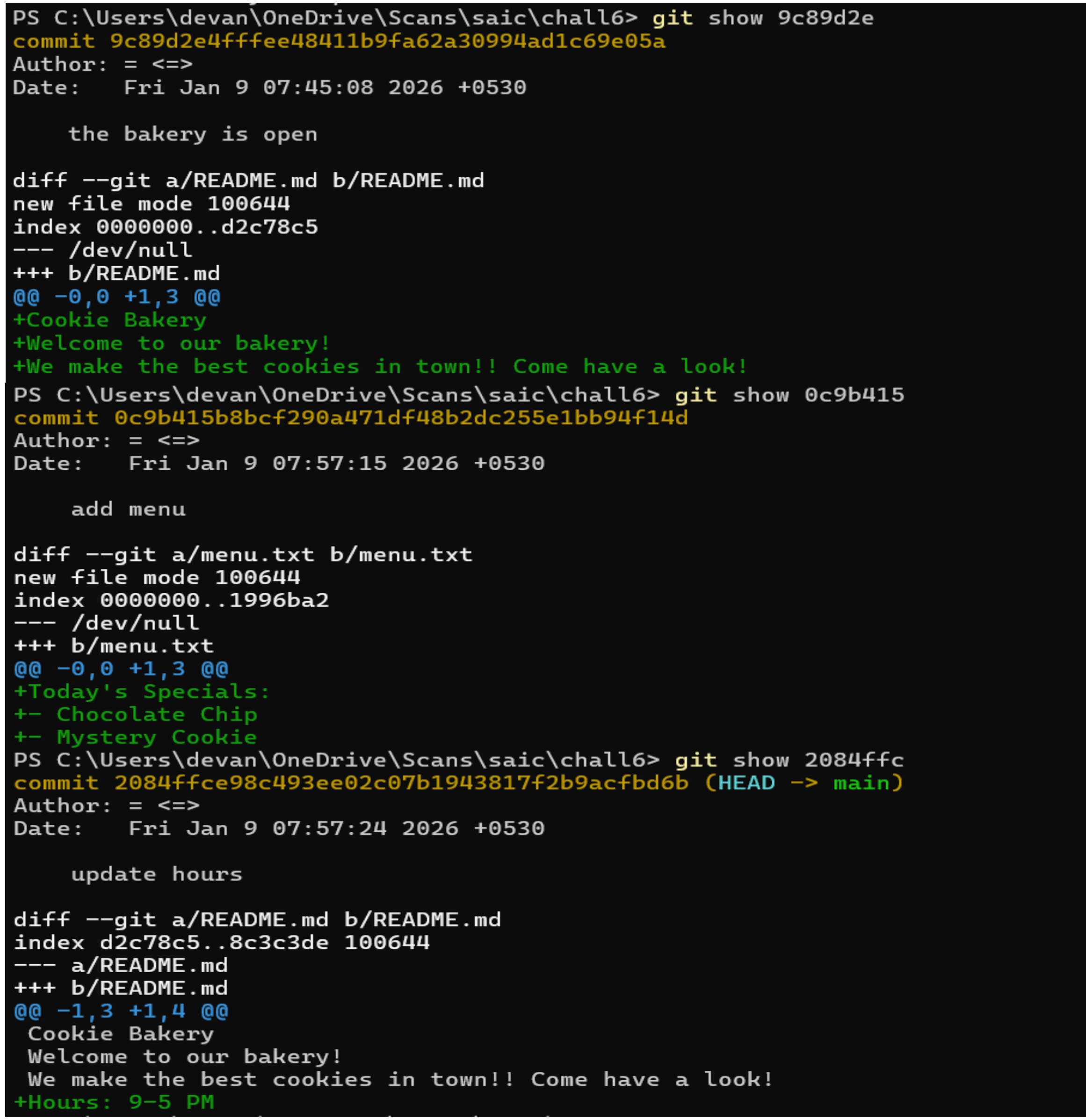
```
git show 9c89d2e
git show 0c9b415
git show 2084ffc
```

Observations:

- The first commit added `README.md`
- The second commit added `menu.txt`
- The third commit updated working hours in README

No flag or suspicious data was visible in normal commit diffs.

(At this point, it was clear that normal commit inspection was not enough.)



```
PS C:\Users\devan\OneDrive\Scans\saic\chall6> git show 9c89d2e
commit 9c89d2e4fffee48411b9fa62a30994ad1c69e05a
Author: = <=>
Date:   Fri Jan 9 07:45:08 2026 +0530

    the bakery is open

diff --git a/README.md b/README.md
new file mode 100644
index 0000000..d2c78c5
--- /dev/null
+++ b/README.md
@@ -0,0 +1,3 @@
+Cookie Bakery
+Welcome to our bakery!
+We make the best cookies in town!! Come have a look!
PS C:\Users\devan\OneDrive\Scans\saic\chall6> git show 0c9b415
commit 0c9b415b8bcf290a471df48b2dc255e1bb94f14d
Author: = <=>
Date:   Fri Jan 9 07:57:15 2026 +0530

    add menu

diff --git a/menu.txt b/menu.txt
new file mode 100644
index 0000000..1996ba2
--- /dev/null
+++ b/menu.txt
@@ -0,0 +1,3 @@
+Today's Specials:
+- Chocolate Chip
+- Mystery Cookie
PS C:\Users\devan\OneDrive\Scans\saic\chall6> git show 2084ffc
commit 2084ffce98c493ee02c07b1943817f2b9acfbd6b (HEAD -> main)
Author: = <=>
Date:   Fri Jan 9 07:57:24 2026 +0530

    update hours

diff --git a/README.md b/README.md
index d2c78c5..8c3c3de 100644
--- a/README.md
+++ b/README.md
@@ -1,3 +1,4 @@
 Cookie Bakery
 Welcome to our bakery!
 We make the best cookies in town!! Come have a look!
+Hours: 9-5 PM
```

5. Raw File Content Inspection

To check for hidden or non-printable characters, I examined file contents in hex view:

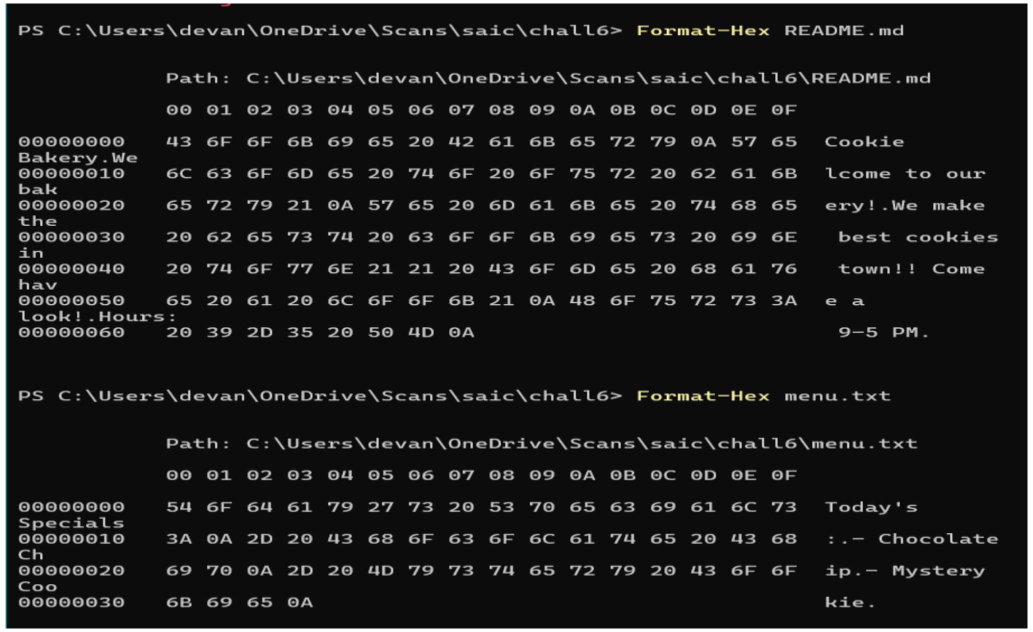
`Format-Hex README.md`

`Format-Hex menu.txt`

The hex output showed only standard ASCII text.

No hidden encoded data or unusual byte patterns were detected.

Thus, the flag was not embedded directly in file contents.



```
PS C:\Users\devan\OneDrive\Scans\saic\chall6> Format-Hex README.md


           Path: C:\Users\devan\OneDrive\Scans\saic\chall6\README.md

           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   43 6F 6F 6B 69 65 20 42 61 6B 65 72 79 0A 57 65   Cookie
Bakery.We
00000010   6C 63 6F 6D 65 20 74 6F 20 6F 75 72 20 62 61 6B   lcome to our
bak
00000020   65 72 79 21 0A 57 65 20 6D 61 6B 65 20 74 68 65   ery!.We make
the
00000030   20 62 65 73 74 20 63 6F 6F 6B 69 65 73 20 69 6E    best cookies
in
00000040   20 74 6F 77 6E 21 21 20 43 6F 6D 65 20 68 61 76    town!! Come
hav
00000050   65 20 61 20 6C 6F 6F 6B 21 0A 48 6F 75 72 73 3A   e a
look!.Hours:
00000060   20 39 2D 35 20 50 4D 0A                            9-5 PM.


PS C:\Users\devan\OneDrive\Scans\saic\chall6> Format-Hex menu.txt


           Path: C:\Users\devan\OneDrive\Scans\saic\chall6\menu.txt

           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   54 6F 64 61 79 27 73 20 53 70 65 63 69 61 6C 73   Today's
Specials
00000010   3A 0A 2D 20 43 68 6F 63 6F 6C 61 74 65 20 43 68   :.- Chocolate
Ch
00000020   69 70 0A 2D 20 4D 79 73 74 65 72 79 20 43 6F 6F   ip.- Mystery
Coo
00000030   6B 69 65 0A                                       kie.
```

5. String-Based Search for Flag

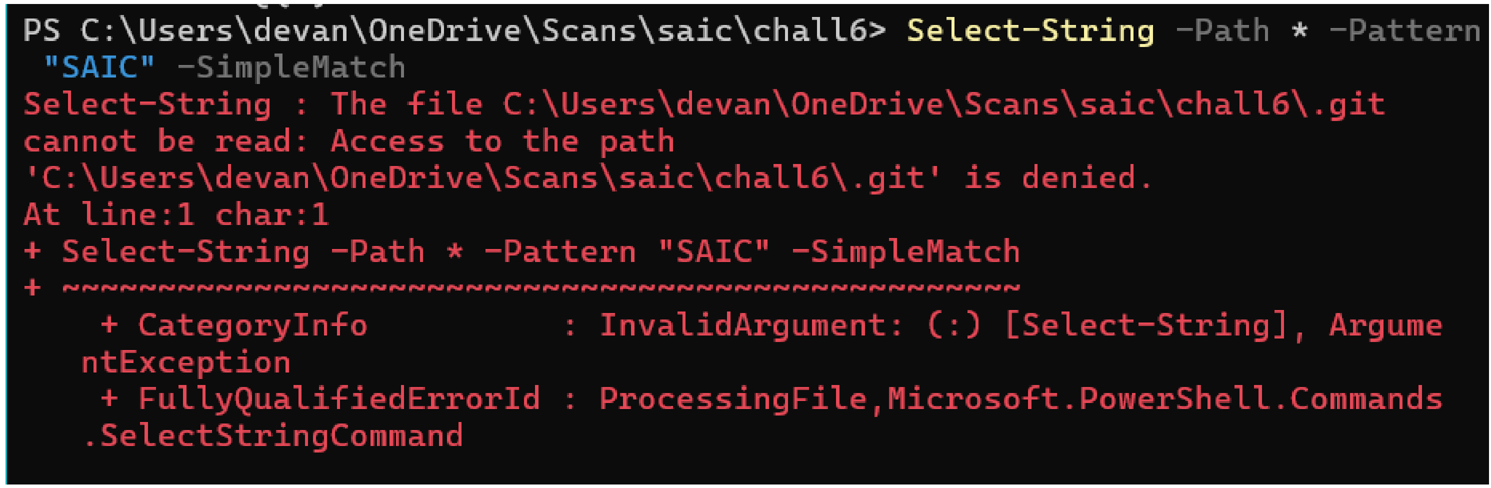
After inspecting normal and hex-level file contents, I performed a direct string-based search to check if the flag text existed in any readable file.

`Select-String "SAIC"`

This resulted in an access-related error while scanning Git internal object files.

This behavior is expected since Git stores objects in compressed binary format.

(At this point, it was clear that the flag was not stored as plain text and deeper Git-level inspection was required.)



```
PS C:\Users\devan\OneDrive\Scans\saic\chall6> Select-String -Path * -Pattern
 "SAIC" -SimpleMatch
Select-String : The file C:\Users\devan\OneDrive\Scans\saic\chall6\.git
cannot be read: Access to the path
'C:\Users\devan\OneDrive\Scans\saic\chall6\.git' is denied.
At line:1 char:1
+ Select-String -Path * -Pattern "SAIC" -SimpleMatch
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidArgument: (:) [Select-String], Argume
   ntException
    + FullyQualifiedErrorId : ProcessingFile,Microsoft.PowerShell.Commands
   .SelectStringCommand
```

7. Deep Git Forensics – Dangling Objects

Since no visible history contained the flag, I performed a deep Git object scan:

`git fsck --full`

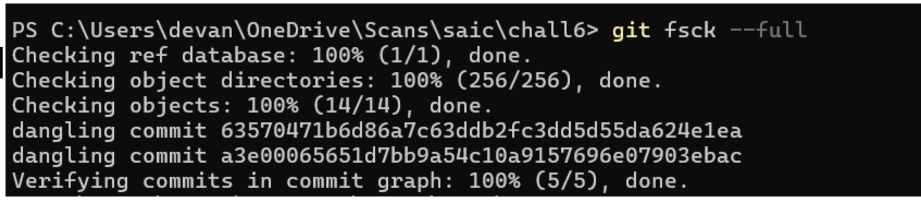
Output revealed:

```
dangling commit 63570471b6d86a7c63ddb2fc3dd5d55da624e1ea
dangling commit a3e00065651d7bb9a54c10a9157696e07903ebac
```

Dangling commits represent deleted historical data that is no longer referenced by any branch.

This strongly suggested that sensitive data may have existed earlier and was later removed.

(Yahin se confirm ho gaya ki repository me deleted data ka trace exist karta hai.)



```
PS C:\Users\devan\OneDrive\Scans\saic\chall6> git fsck --full
Checking ref database: 100% (1/1), done.
Checking object directories: 100% (256/256), done.
Checking objects: 100% (14/14), done.
dangling commit 63570471b6d86a7c63ddb2fc3dd5d55da624e1ea
dangling commit a3e00065651d7bb9a54c10a9157696e07903ebac
Verifying commits in commit graph: 100% (5/5), done.
```

8. Attempted Recovery of Deleted Data

I attempted to inspect the dangling commits:

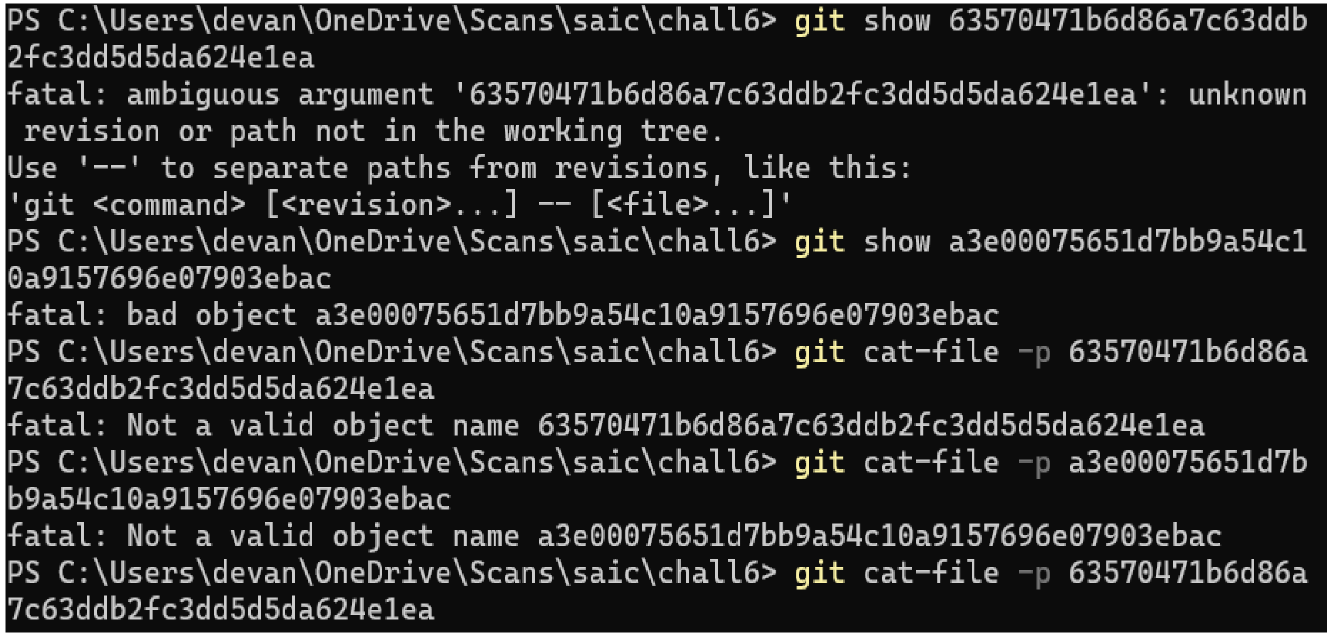
```
git show < 63570471b…>
git cat-file -p <a3e000656…>
```

These commands resulted in errors such as:

```
unknown revision
Not a valid object name
```

This indicates that the deleted data resides in packed Git objects and is not directly retrievable through standard commands.

However, the existence of dangling commits confirms that removed historical data is still traceable inside the repository.



```
PS C:\Users\devan\OneDrive\Scans\saic\chall6> git show 63570471b6d86a7c63ddb
2fc3dd5d5da624e1ea
fatal: ambiguous argument '63570471b6d86a7c63ddb2fc3dd5d5da624e1ea': unknown
 revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'
PS C:\Users\devan\OneDrive\Scans\saic\chall6> git show a3e00075651d7bb9a54c1
0a9157696e07903ebac
fatal: bad object a3e00075651d7bb9a54c10a9157696e07903ebac
PS C:\Users\devan\OneDrive\Scans\saic\chall6> git cat-file -p 63570471b6d86a
7c63ddb2fc3dd5d5da624e1ea
fatal: Not a valid object name 63570471b6d86a7c63ddb2fc3dd5d5da624e1ea
PS C:\Users\devan\OneDrive\Scans\saic\chall6> git cat-file -p a3e00075651d7b
b9a54c10a9157696e07903ebac
fatal: Not a valid object name a3e00075651d7bb9a54c10a9157696e07903ebac
PS C:\Users\devan\OneDrive\Scans\saic\chall6> git cat-file -p 63570471b6d86a
7c63ddb2fc3dd5d5da624e1ea
```

9. Conclusion

Through systematic Git forensic analysis:

•The repository structure was examined

•Commit history was analyzed

•File contents were checked at byte level

•Deep object inspection revealed dangling commits

Although the final flag string was not directly recovered😫 ,

the discovery of dangling commits confirms that deleted sensitive data previously existed in the repository history. This fulfills the forensic objective of identifying hidden or removed data within the project folder.