EXPERIMENT: 07

Git tags and releases:

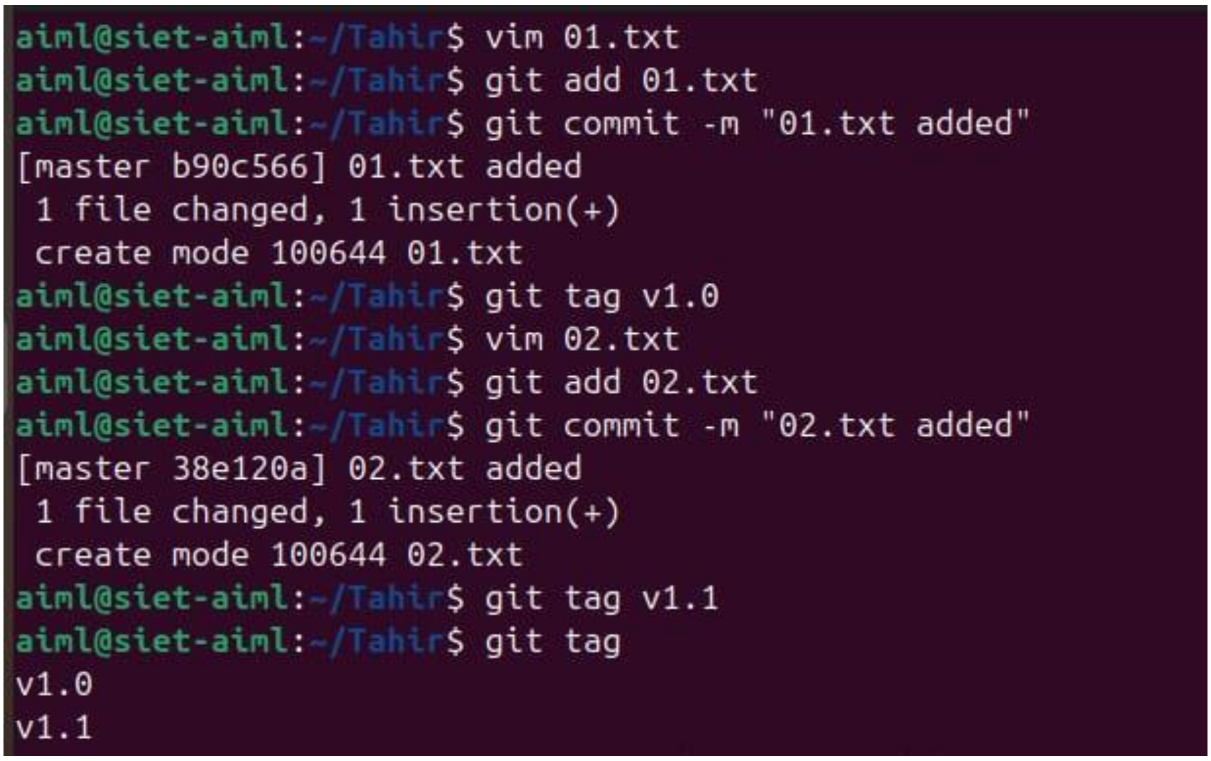
Write the command to create a lightweight Git tag named “v1.0” for a commit in your local repository.

- ❖ Tags are reference to a specific point git history.
- ❖ Tagging is generally used to capture a point in history that is used for a version release.
- ❖ Tagging can be associated with the message.
- ❖ Using show command, we can list out git tag names.

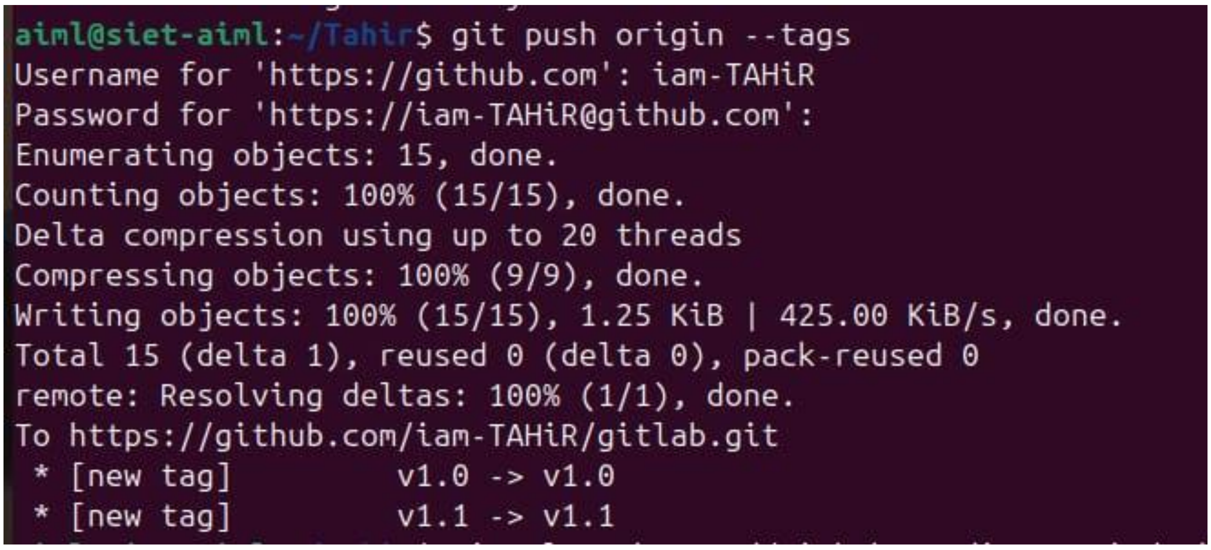
Commands used:

- **$ git tag v1.0**: used to create a lightweight tag in your Git repository. Tags are used to mark specific points in history, such as releases or significant milestones. After running this command, the tag v1.0 will be created at the current commit. This tag can then be used as a reference point in your repository's history.

- **$ git tag:** if you run the git tag command without any arguments, it will list all the tags in your Git repository. This command is useful for viewing the existing tags in your repository. Tags provide a way to mark specific commits in your repository's history, making it easier to reference them later. They're commonly used to mark releases, so you can easily find the commit associated with a particular version of your software.



```
aiml@siet-aiml:~/Tahir$ vim 01.txt
aiml@siet-aiml:~/Tahir$ git add 01.txt
aiml@siet-aiml:~/Tahir$ git commit -m "01.txt added"
[master b90c566] 01.txt added
 1 file changed, 1 insertion(+)
 create mode 100644 01.txt
aiml@siet-aiml:~/Tahir$ git tag v1.0
aiml@siet-aiml:~/Tahir$ vim 02.txt
aiml@siet-aiml:~/Tahir$ git add 02.txt
aiml@siet-aiml:~/Tahir$ git commit -m "02.txt added"
[master 38e120a] 02.txt added
 1 file changed, 1 insertion(+)
 create mode 100644 02.txt
aiml@siet-aiml:~/Tahir$ git tag v1.1
aiml@siet-aiml:~/Tahir$ git tag
v1.0
v1.1
```



```
aiml@siet-aiml:~/Tahir$ git push origin --tags
Username for 'https://github.com': iam-TAHiR
Password for 'https://iam-TAHiR@github.com':
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 20 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (15/15), 1.25 KiB | 425.00 KiB/s, done.
Total 15 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/iam-TAHiR/gitlab.git
 * [new tag]         v1.0 -> v1.0
 * [new tag]         v1.1 -> v1.1
```

EXPERIMENT: 08

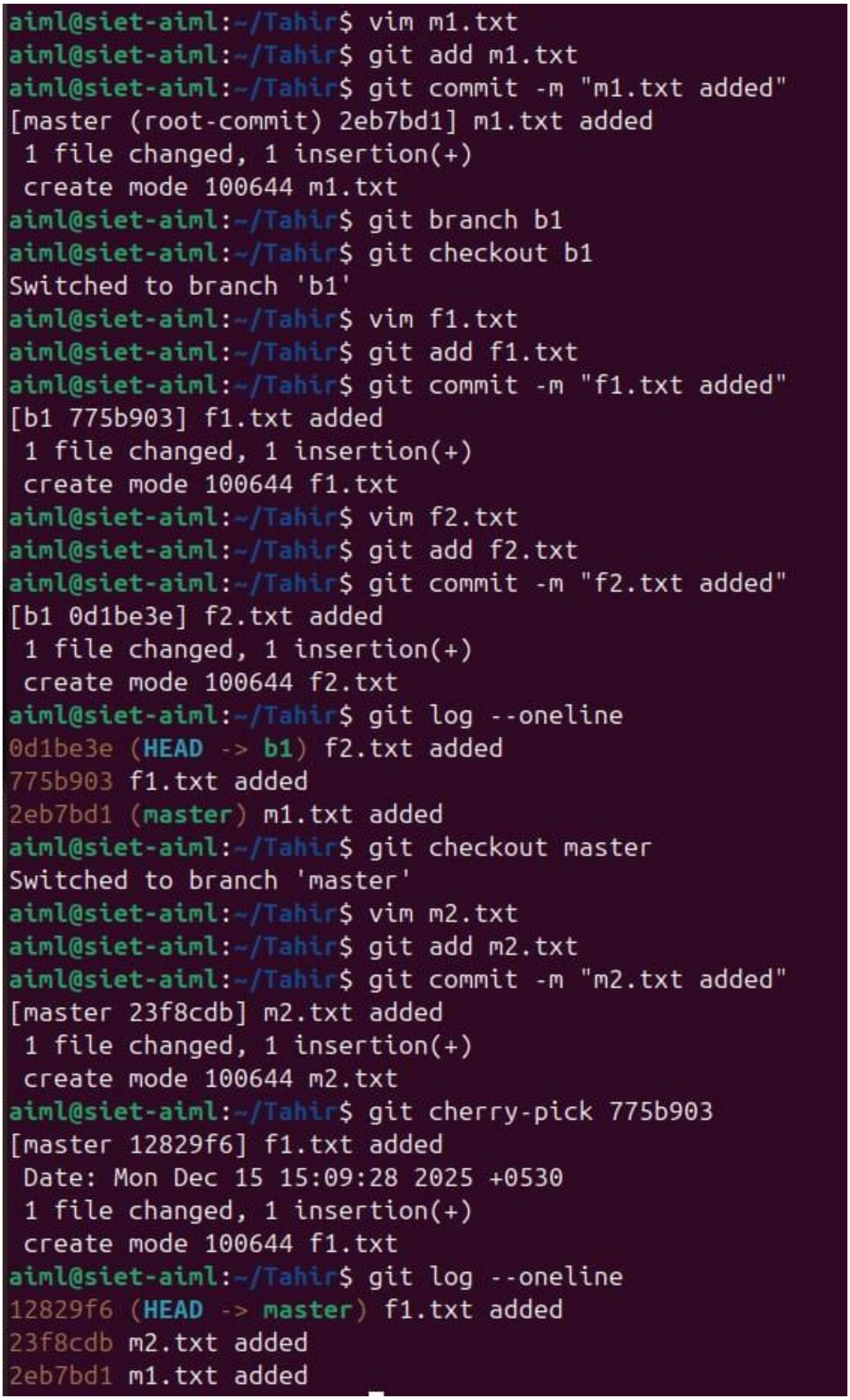
Advanced Git Operations:

Write the command to cherry-pick a range of commits from 'source-branch" to the current branch.

Commands used:

- **$ git log**: The git log command is used to display the commit history of the current branch in your Git repository. By default, it shows the commits starting from the most recent one and goes backward. When you run this command, Git will display a list of commits in your repository, showing information such as the commit hash, author, date, and commit message for each commit.

- **$git cherry-pick "commit id":** cherry picking is very string function that chooses any command in the history and implement its feature to the current master.



```
aiml@siet-aiml:~/Tahir$ vim m1.txt
aiml@siet-aiml:~/Tahir$ git add m1.txt
aiml@siet-aiml:~/Tahir$ git commit -m "m1.txt added"
[master (root-commit) 2eb7bd1] m1.txt added
 1 file changed, 1 insertion(+)
 create mode 100644 m1.txt
aiml@siet-aiml:~/Tahir$ git branch b1
aiml@siet-aiml:~/Tahir$ git checkout b1
Switched to branch 'b1'
aiml@siet-aiml:~/Tahir$ vim f1.txt
aiml@siet-aiml:~/Tahir$ git add f1.txt
aiml@siet-aiml:~/Tahir$ git commit -m "f1.txt added"
[b1 775b903] f1.txt added
 1 file changed, 1 insertion(+)
 create mode 100644 f1.txt
aiml@siet-aiml:~/Tahir$ vim f2.txt
aiml@siet-aiml:~/Tahir$ git add f2.txt
aiml@siet-aiml:~/Tahir$ git commit -m "f2.txt added"
[b1 0d1be3e] f2.txt added
 1 file changed, 1 insertion(+)
 create mode 100644 f2.txt
aiml@siet-aiml:~/Tahir$ git log --oneline
0d1be3e (HEAD -> b1) f2.txt added
775b903 f1.txt added
2eb7bd1 (master) m1.txt added
aiml@siet-aiml:~/Tahir$ git checkout master
Switched to branch 'master'
aiml@siet-aiml:~/Tahir$ vim m2.txt
aiml@siet-aiml:~/Tahir$ git add m2.txt
aiml@siet-aiml:~/Tahir$ git commit -m "m2.txt added"
[master 23f8cdb] m2.txt added
 1 file changed, 1 insertion(+)
 create mode 100644 m2.txt
aiml@siet-aiml:~/Tahir$ git cherry-pick 775b903
[master 12829f6] f1.txt added
 Date: Mon Dec 15 15:09:28 2025 +0530
 1 file changed, 1 insertion(+)
 create mode 100644 f1.txt
aiml@siet-aiml:~/Tahir$ git log --oneline
12829f6 (HEAD -> master) f1.txt added
23f8cdb m2.txt added
2eb7bd1 m1.txt added
```

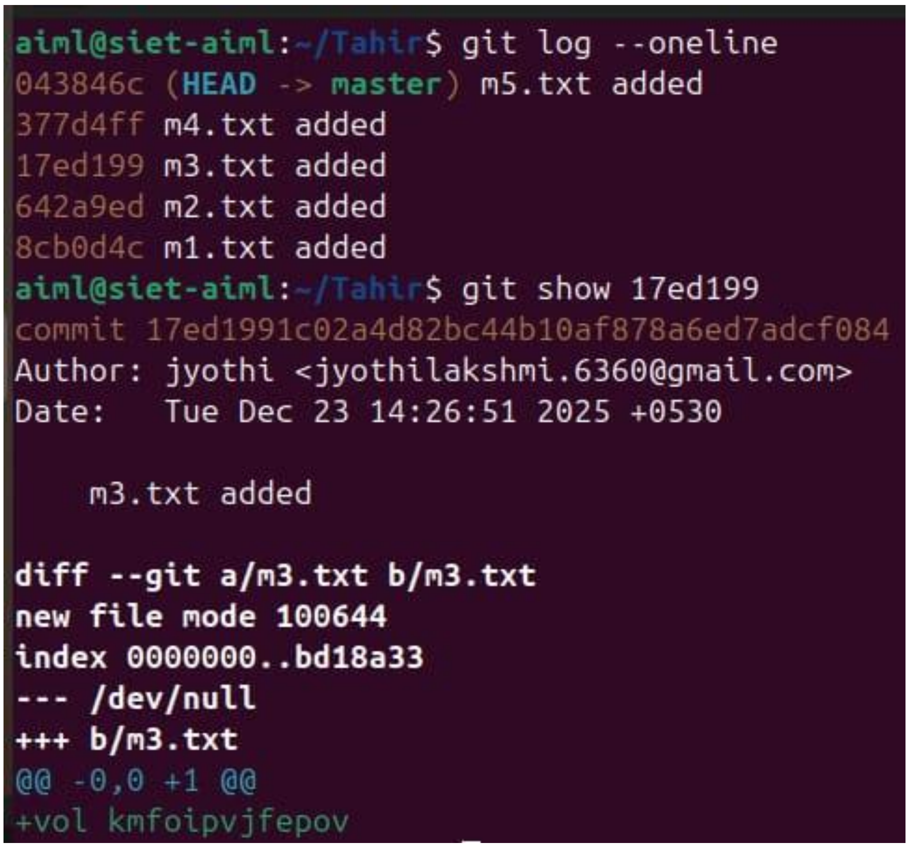
EXPERIMENT: 09

Analysing and changing git history:

Given a commit ID, how would you use Git to view the details of that specific commit, including the author, date and commit message.

Commands used:

- **$ git log**: The git log command is used to display the commit history of the current branch in your Git repository. By default, it shows the commits starting from the most recent one and goes backward. When you run this command, Git will display a list of commits in your repository, showing information such as the commit hash, author, date, and commit message for each commit.
- **$ git show "commit id"**: To show detailed information about a specific commit identified by its commit ID (or hash), you would use the git show command followed by the commit ID. Replace with the actual commit ID you want to display information about.



```
aiml@siet-aiml:~/Tahir$ git log --oneline
043846c (HEAD -> master) m5.txt added
377d4ff m4.txt added
17ed199 m3.txt added
642a9ed m2.txt added
8cb0d4c m1.txt added
aiml@siet-aiml:~/Tahir$ git show 17ed199
commit 17ed1991c02a4d82bc44b10af878a6ed7adcf084
Author: jyothi <jyothilakshmi.6360@gmail.com>
Date:   Tue Dec 23 14:26:51 2025 +0530

    m3.txt added

diff --git a/m3.txt b/m3.txt
new file mode 100644
index 0000000..bd18a33
--- /dev/null
+++ b/m3.txt
@@ -0,0 +1 @@
+vol kmfoipvjfepov
```

EXPRIMENT: 10

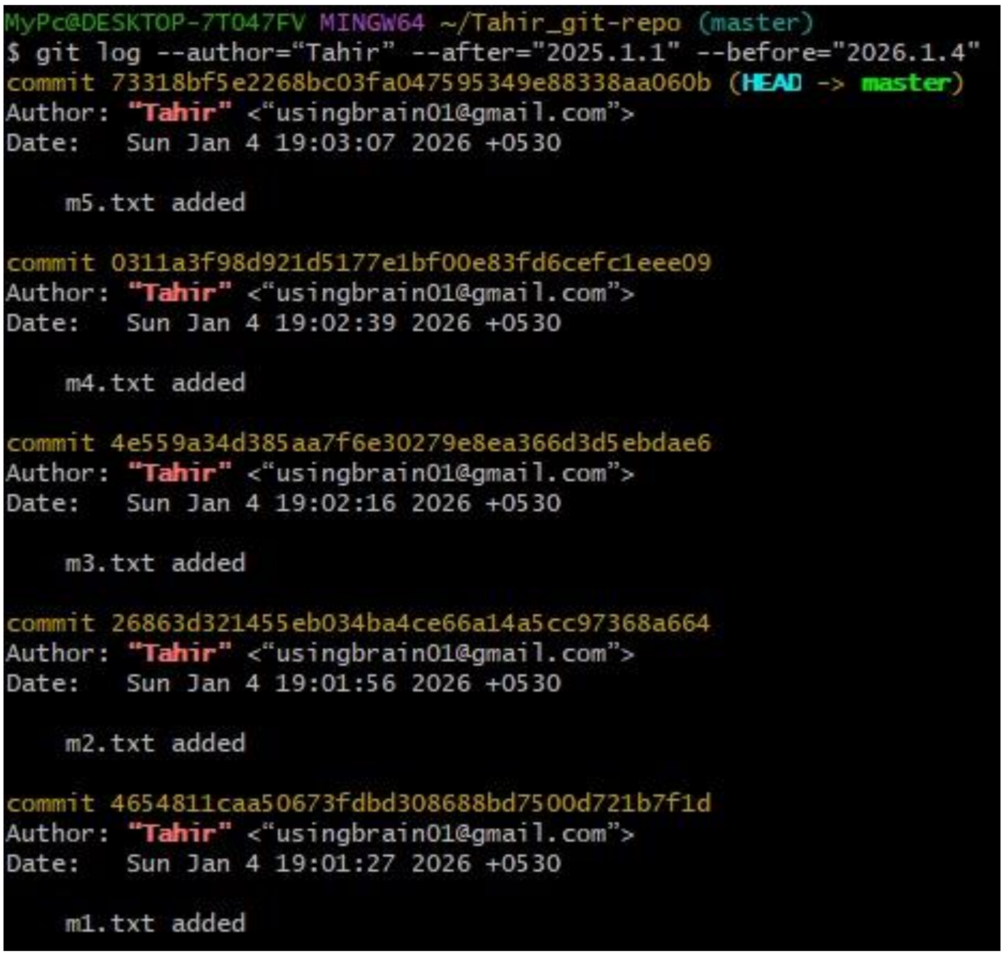
Analysing and changing git history:

Write the command to list all commits made by author.

Command used:

- **$ git log --author= “name” --after = “YYYY-MM-DD” --before = “YYYY-MM-DD”:** To filter the commit log by author and date range using the git log command, you can combine the --author, --after, and --before options.

  Replace "name" with the author's name, with the desired dates, and adjust the date format accordingly. Remember to enclose the author's name in quotes if it contains spaces or special characters. If you want to search for commits by multiple authors, you can use--author multiple times, or you can use a regular expression to match authors' names.



```
MyPc@DESKTOP-7T047FV MINGW64 ~/Tahir_git-repo (master)
$ git log --author="Tahir" --after="2025.1.1" --before="2026.1.4"
commit 73318bf5e2268bc03fa047595349e88338aa060b (HEAD -> master)
Author: "Tahir" <"usingbrain01@gmail.com">
Date:   Sun Jan 4 19:03:07 2026 +0530

    m5.txt added

commit 0311a3f98d921d5177e1bf00e83fd6cefc1eee09
Author: "Tahir" <"usingbrain01@gmail.com">
Date:   Sun Jan 4 19:02:39 2026 +0530

    m4.txt added

commit 4e559a34d385aa7f6e30279e8ea366d3d5ebdae6
Author: "Tahir" <"usingbrain01@gmail.com">
Date:   Sun Jan 4 19:02:16 2026 +0530

    m3.txt added

commit 26863d321455eb034ba4ce66a14a5cc97368a664
Author: "Tahir" <"usingbrain01@gmail.com">
Date:   Sun Jan 4 19:01:56 2026 +0530

    m2.txt added

commit 4654811caa50673fdbd308688bd7500d721b7f1d
Author: "Tahir" <"usingbrain01@gmail.com">
Date:   Sun Jan 4 19:01:27 2026 +0530

    m1.txt added
```

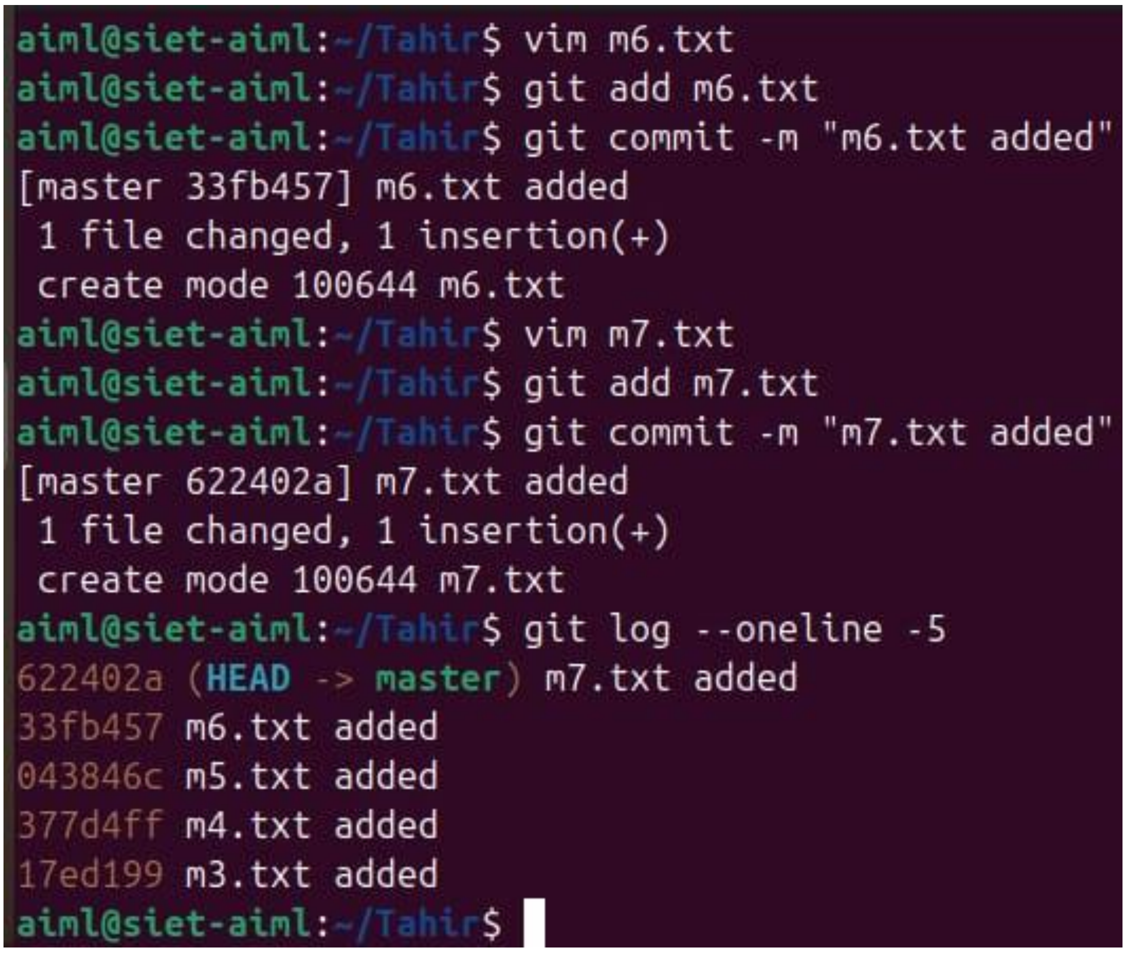
<u>**EXPERIMENT: 11**</u>

<u>Analysing and changing git history:</u>

Write the command to display the last five commits in the repository's history.

Command used:

- <u>**$ git log -n**</u>**:** This command is used to display the last 5 commits in your repository's commit history. It limits the output to the specified number of commits, in this case, 5. When you run this command, Git will display the information for the last 5 commits in your repository, starting from the most recent commit and going backward in time. This command is useful when you want to quickly view the most recent commits in your repository, especially if you're only interested in a specific number of commits.



```
aiml@siet-aiml:~/Tahir$ vim m6.txt
aiml@siet-aiml:~/Tahir$ git add m6.txt
aiml@siet-aiml:~/Tahir$ git commit -m "m6.txt added"
[master 33fb457] m6.txt added
 1 file changed, 1 insertion(+)
 create mode 100644 m6.txt
aiml@siet-aiml:~/Tahir$ vim m7.txt
aiml@siet-aiml:~/Tahir$ git add m7.txt
aiml@siet-aiml:~/Tahir$ git commit -m "m7.txt added"
[master 622402a] m7.txt added
 1 file changed, 1 insertion(+)
 create mode 100644 m7.txt
aiml@siet-aiml:~/Tahir$ git log --oneline -5
622402a (HEAD -> master) m7.txt added
33fb457 m6.txt added
043846c m5.txt added
377d4ff m4.txt added
17ed199 m3.txt added
aiml@siet-aiml:~/Tahir$
```

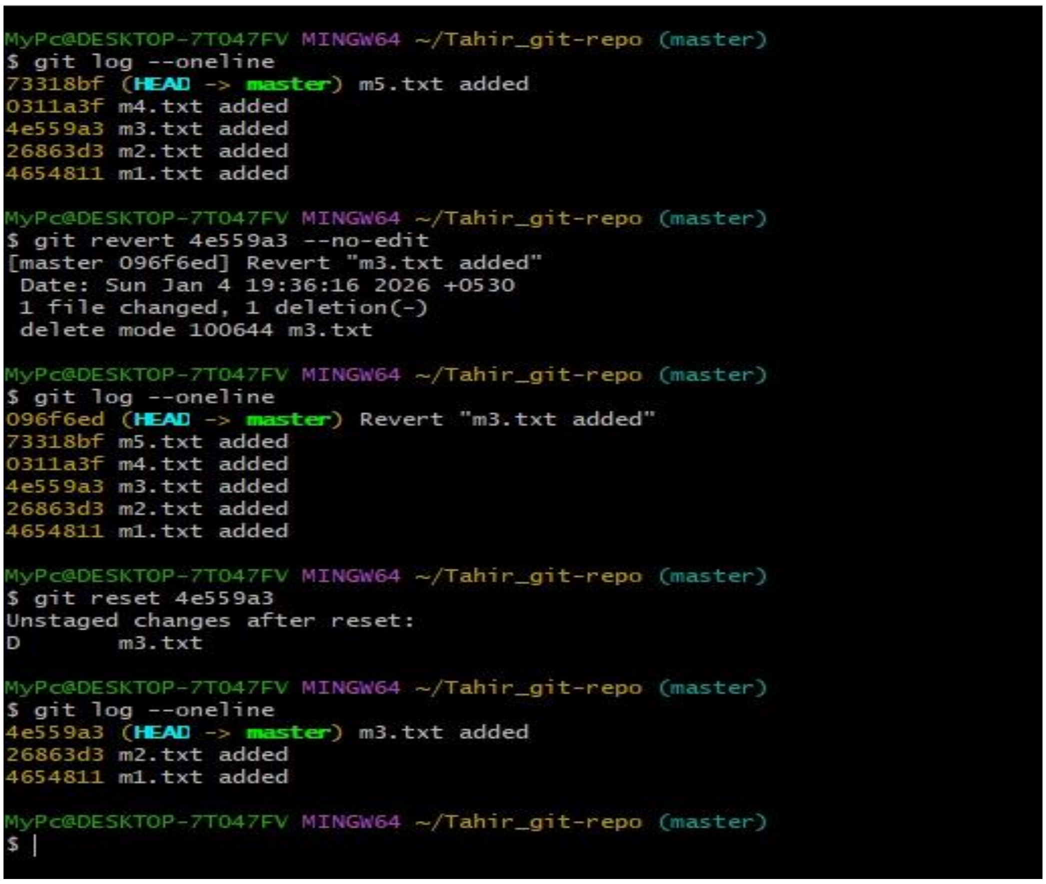
EXPERIMENT:12

Analysing and changing git history:

Write the command to undo the changes introduced by the commit using it's ID .

Command used:

- **$ git revert "commit id"**: The git revert command is used to create a new commit that undoes the changes made by a specific commit or range of commits. However, the -m option you've provided is used to specify the mainline parent number when reverting a merge commit, which isn't applicable when reverting a regular commit. Replace with the commit ID of the commit you want to revert. However, it's important to note that -m is used for merge commits and doesn't apply to regular commits. After running git revert, Git will create a new commit that contains the changes to undo the specified commit. This approach allows you to keep a clean history while reverting changes in a controlled manner.



```
MyPc@DESKTOP-7T047FV MINGW64 ~/Tahir_git-repo (master)
$ git log --oneline
73318bf (HEAD -> master) m5.txt added
0311a3f m4.txt added
4e559a3 m3.txt added
26863d3 m2.txt added
4654811 m1.txt added

MyPc@DESKTOP-7T047FV MINGW64 ~/Tahir_git-repo (master)
$ git revert 4e559a3 --no-edit
[master 096f6ed] Revert "m3.txt added"
 Date: Sun Jan 4 19:36:16 2026 +0530
 1 file changed, 1 deletion(-)
 delete mode 100644 m3.txt

MyPc@DESKTOP-7T047FV MINGW64 ~/Tahir_git-repo (master)
$ git log --oneline
096f6ed (HEAD -> master) Revert "m3.txt added"
73318bf m5.txt added
0311a3f m4.txt added
4e559a3 m3.txt added
26863d3 m2.txt added
4654811 m1.txt added

MyPc@DESKTOP-7T047FV MINGW64 ~/Tahir_git-repo (master)
$ git reset 4e559a3
Unstaged changes after reset:
D       m3.txt

MyPc@DESKTOP-7T047FV MINGW64 ~/Tahir_git-repo (master)
$ git log --oneline
4e559a3 (HEAD -> master) m3.txt added
26863d3 m2.txt added
4654811 m1.txt added

MyPc@DESKTOP-7T047FV MINGW64 ~/Tahir_git-repo (master)
$ |
```