VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”,Machhe,Belagavi,Karnataka-590018

Lab Experiment Record

Project Management with Git [BCSL358C]

Submitted In partialfulfillment towards AEC of 3rdsemesterof

Bachelorof Engineering
in
ComputerScienceandEngineering
(ArtificialIntelligence&MachineLearning)

Submittedby

TELUGU SHIVANI
4GW24CI057

DEPARTMENTOF CSE(ArtificialIntelligence&MachineLearning)

GSSS INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN

(AffiliatedtoVTU,Belagavi,ApprovedbyAICTE,NewDelhi&Govt.ofKarnataka)

K.R.SROAD,METAGALLI,MYSURU-570016,KARNATAKA

(AccreditedbyNAAC)

2025-2026

INDEX

SYLLABUS

1. **Setting Up and Basic Commands**
   Initialize a new Git repository in a directory. Create a new file and add it to the staging area and commit the changes with an appropriate commit message.

2. **Creating and Managing Branches**
   Create a new branch named "feature-branch." Switch to the "master" branch. Merge the "feature-branch" into "master."

3. **Creating and Managing Branches**
   Write the commands to stash your changes, switch branches, and then apply the stashed changes.

4. **Collaboration and Remote Repositories**
   Clone a remote Git repository to your local machine.

5. **Collaboration and Remote Repositories**
   Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch.

6. **Collaboration and Remote Repositories**
   Write the command to merge "feature-branch" into "master" while providing a custom commit message for the merge.

7. **Git Tags and Releases**
   Write the command to create a lightweight Git tag named "v1.0" for a commit in your local repository.

8. **Advanced Git Operations**
   Write the command to cherry-pick a range of commits from "source-branch" to the current.

9. **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?

10. Analysing and Changing Git History
Write the command to list all commits made by the author "JohnDoe" between "2023-01-01" and "2023-12-31."

11. Analysing and Changing Git History
Write the command to display the last five commits in the repository's history.

12. Analysing and Changing Git History
Write the command to undo the changes introduced by the commit with the ID "abc123".

Git Life Cycle

The Git lifecycle refers to the typical sequence of actions and steps you take when using Git to manage your source code and collaborate with others. Here's an overview of the Git lifecycle:

1. **Initializing a Repository**:
To start using Git, you typically initialize a new repository (or repo) in your project directory. This is done with the command git init.

2. **Working Directory**:
Your project files exist in the working directory. These are the files you are actively working on.

3. **Staging**:
Before you commit changes, you need to stage them. Staging allows you to select which changes you want to include in the next commit. You use the git add command to stage changes selectively or all at once with git add.
.
4. **Committing**:
After you've staged your changes, you commit them with a message explaining what you've done. Commits create snapshots of your project at that point in time. You use the git commit command to make commits, like git commit –m "Add new feature".

5. **Local Repository**:
Commits are stored in your local repository. Your project's version history is preserved there.

6. **Branching**:
Git encourages branching for development. You can create branches to work on new features, bug fixes, or experiments without affecting the main codebase. Use the git branch and git checkout commands for branching.

7. **Merging**:

After you've completed work in a branch and want to integrate it into the main codebase, you perform a merge. Merging combines the changes from one branch into another. Use the git merge command.

8. Remote Repository:
For collaboration, you can work with remote repositories hosted on servers like GitHub, GitLab, or Bitbucket. These repositories serve as a central hub for sharing code.

9. Pushing:
To share your local commits with a remote repository, you push them using the git push command. This updates the remote repository with your changes.

10.Pulling:
To get changes made by others in the remote repository, you pull them to your local repository with the git pull command. This ensures that your local copy is up to date.

11.Conflict Resolution:
Conflicts can occur when multiple people make changes to the same part of a file. Git will inform you of conflicts, and you must resolve them by editing the affected files manually.

12.Collaboration:
Developers can collaborate by pushing, pulling, and making pull requests in a shared remote repository. Collaboration tools like pull requests are commonly used on platforms like GitHub and GitLab.

13.Tagging and Releases:
You can create tags to mark specific points in the project's history, such as version releases. Tags are useful for identifying significant milestones.

14.Continuous Cycle:
The Git lifecycle continues as you repeat these steps over time to manage the ongoing development and evolution of your project. This cycle supports collaborative and agile software development.

Git Commands List

Git is a popular version control system used for tracking changes in software development projects. Here's a list of common Git commands along with brief explanations:

1. **git init**: Initializes a new Git repository in the current directory.
2. **git clone <repository URL>**: Creates a copy of a remote repository on your local machine.
3. **git add <file>**: Stages a file to be committed, marking it for tracking in the next commit.
4. **git commit -m "message"**: Records the changes you've staged with a descriptive commit message.
5. **git status**: Shows the status of your working directory and the files that have been modified or staged.
6. **git log**: Displays a log of all previous commits, including commit hashes, authors, dates, and commit messages.
7. **git diff**: Shows the differences between the working directory and the last committed version.
8. **git branch**: Lists all branches in the repository and highlights the currently checkedout branch.
9. **git branch <branchname>**: Creates a new branch with the specified name.
10. **git checkout <branchname>**: Switches to a different branch.
11. **git merge <branchname>**: Merges changes from the specified branch into the currently checked-out branch.
12. **git pull**: Fetches changes from a remote repository and merges them into the current branch.
13. **git push**: Pushes your local commits to a remote repository.
14. **git remote**: Lists the remote repositories that your local repository is connected to.
15. **git fetch**: Retrieves changes from a remote repository without merging them.
16. **git reset <file>**: Unstages a file that was previously staged for commit.
17. **git reset --hard <commit>**: Resets the branch to a specific commit, discarding all changes after that commit.
18. **git stash**: Temporarily saves your changes to a "stash" so you can switch branches without committing or losing your work.

19. **git tag**: Lists and manages tags (usually used for marking specific points in history, like releases).
20. **git blame <file>**: Shows who made each change to a file and when.
21. **git rm <file>**: Removes a file from both your working directory and the Git repository.
22. **git mv <oldfile> <newfile>**: Renames a file and stages the change.

These are some of the most common Git commands, but Git offers a wide range of features and options for more advanced usage. You can use git --help followed by the command name
to get more information about any specific command, e.g., git help commit.

Global Configuration:

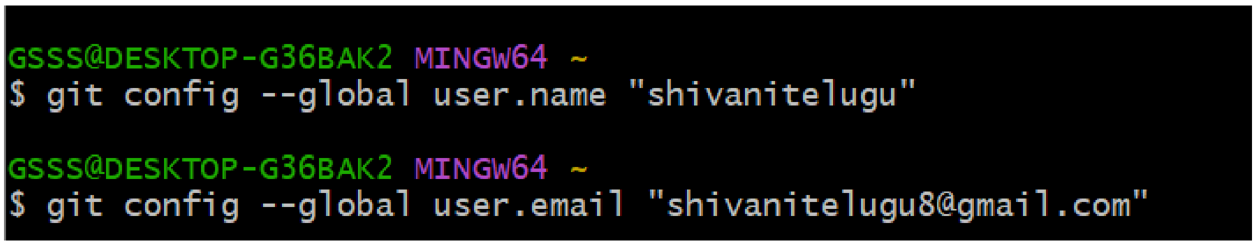
Global configuration is specific to your user account and applies to all Git repositories
on your computer. This is where you usually set your name and email.
To set global configuration, you can use the git config command with the --global flag.

For example:

```
$ git config --global user.name "Your Name"
$ git config --global user.email your.email@example.com
```

You can also view your global Git configuration by using:

```
$ git config --global –list
```



```
GSSS@DESKTOP-G36BAK2 MINGW64 ~
$ git config --global user.name "shivanitelugu"

GSSS@DESKTOP-G36BAK2 MINGW64 ~
$ git config --global user.email "shivanitelugu8@gmail.com"
```

Experiments On Project Management with Git

Experiment 1:

Setting Up and Basic Commands:

Initialize a new Git repository in a directory. Create a new file and add it to the staging area and commit the changes with an appropriate commit message.

Step 1: Create a new directory

```
mkdir git-exp1
cd git-exp1
```

Step 2: Initialize a new Git repository

```
git  init
```

Step 3: Create a new file

```
touch file1.txt
```

Step 4: Check repository status

```
git status
```

Step 5: Add file to staging area

```
git add file1.txt
```

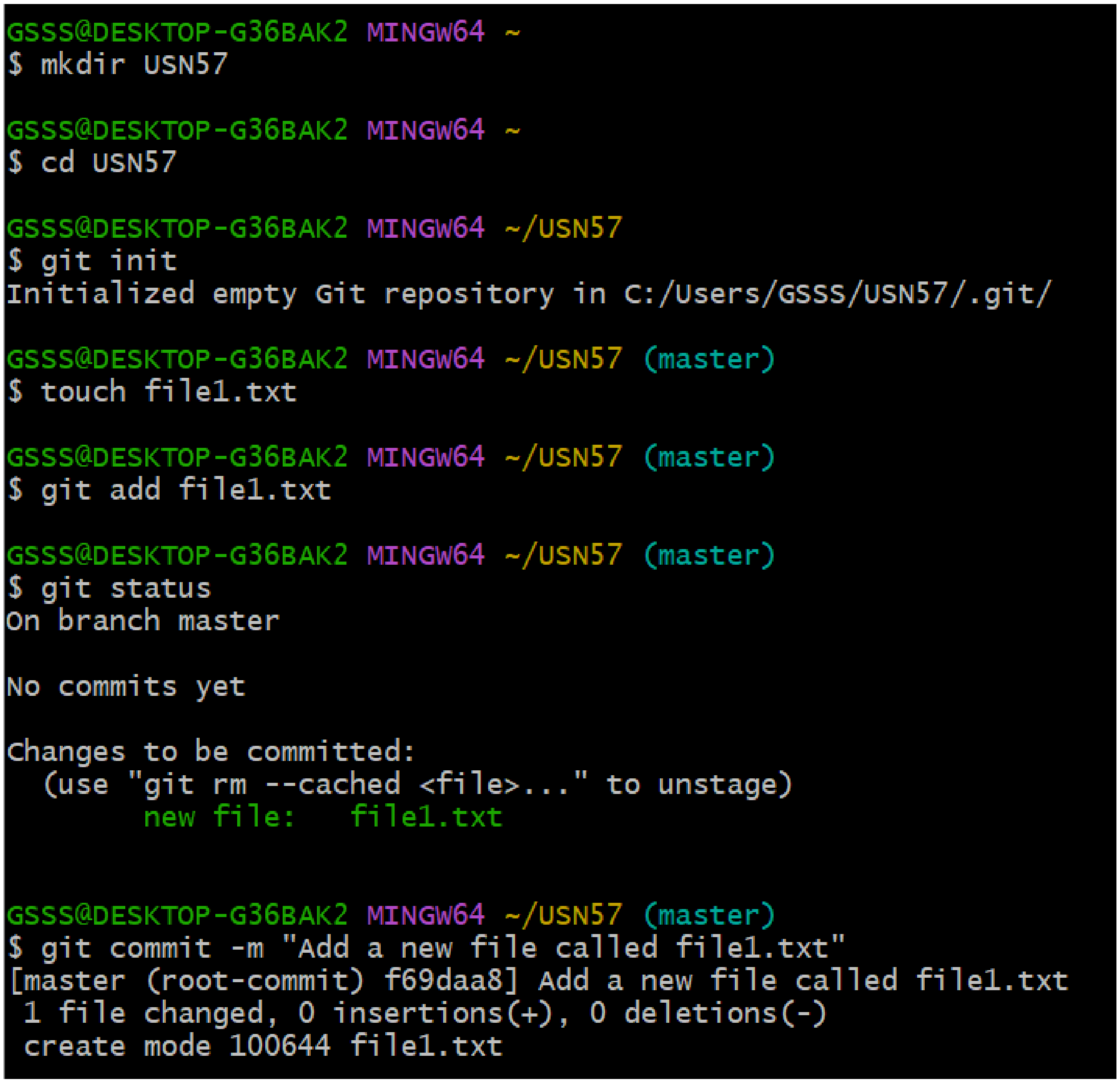
Step 6: Verify staged file

```
git status
```

Step 6: Verify staged file

```
git status
```

EXPECTED OUTPUT:



```
GSSS@DESKTOP-G36BAK2 MINGW64 ~
$ mkdir USN57

GSSS@DESKTOP-G36BAK2 MINGW64 ~
$ cd USN57

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57
$ git init
Initialized empty Git repository in C:/Users/GSSS/USN57/.git/

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ touch file1.txt

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ git add file1.txt

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt


GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ git commit -m "Add a new file called file1.txt"
[master (root-commit) f69daa8] Add a new file called file1.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1.txt
```

Experiment 2:

Creating and Managing Branches:

Create a new branch named "feature-branch." Switch to the "master" branch.
Merge the "feature-branch" into "master."

Step 1: Create a new branch named feature-branch

```
git branch feature-branch
```

Step 2: Switch to feature-branch

```
git checkout feature-branch
```

Step 3: Make changes in feature-branch (example)

```
echo "This is a feature update" >> file1.txt
git add file1.txt
git commit -m "Added feature update"
```

Step 4: Switch back to master branch

```
git checkout master
```

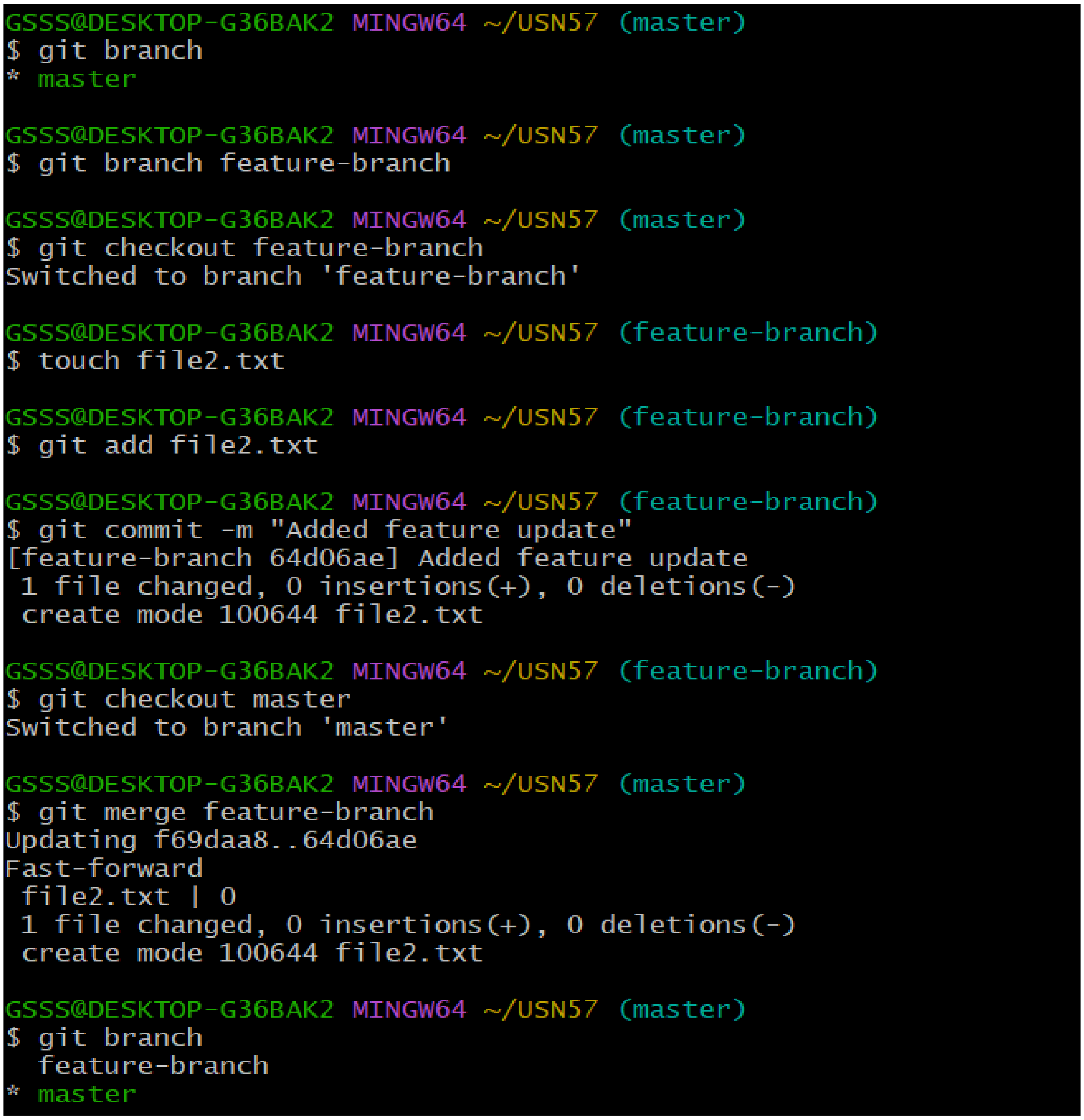
Step 5: Merge feature-branch into master

```
git merge feature-branch
```

Step 6: Verify branches

git branch

EXPECTED OUTPUT:



```
GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ git branch
* master

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ git branch feature-branch

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ git checkout feature-branch
Switched to branch 'feature-branch'

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ touch file2.txt

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git add file2.txt

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git commit -m "Added feature update"
[feature-branch 64d06ae] Added feature update
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file2.txt

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git checkout master
Switched to branch 'master'

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ git merge feature-branch
Updating f69daa8..64d06ae
Fast-forward
 file2.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file2.txt

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ git branch
  feature-branch
* master
```

Experiment 3:

Creating and Managing Branches:

Write the commands to stash your changes, switch branches, and then apply the stashed changes.

Step 1: Check current changes

```
git status
```

Step 2: Stash the changes

```
git stash
```

Step 3: Switch to another branch

```
git checkout master
```

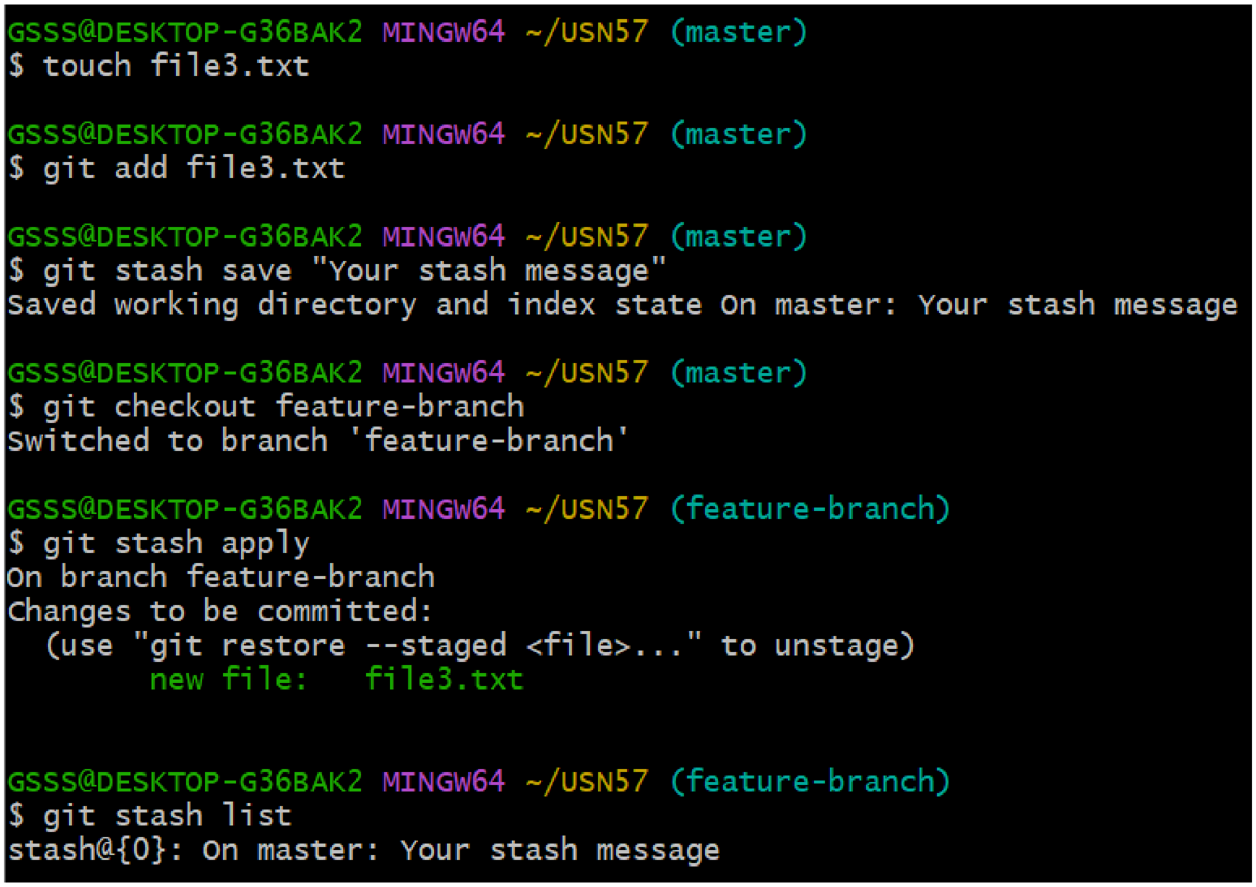
OR

```
git switch master
```

Step 4: Apply the stashed changes

```
git stash apply
```

EXPECTED OUTPUT:



```
GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ touch file3.txt

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ git add file3.txt

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ git stash save "Your stash message"
Saved working directory and index state On master: Your stash message

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (master)
$ git checkout feature-branch
Switched to branch 'feature-branch'

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git stash apply
On branch feature-branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file3.txt


GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git stash list
stash@{0}: On master: Your stash message
```

Experiment 4:

Collaboration and Remote Repositories:

Clone a remote Git repository to your local machine

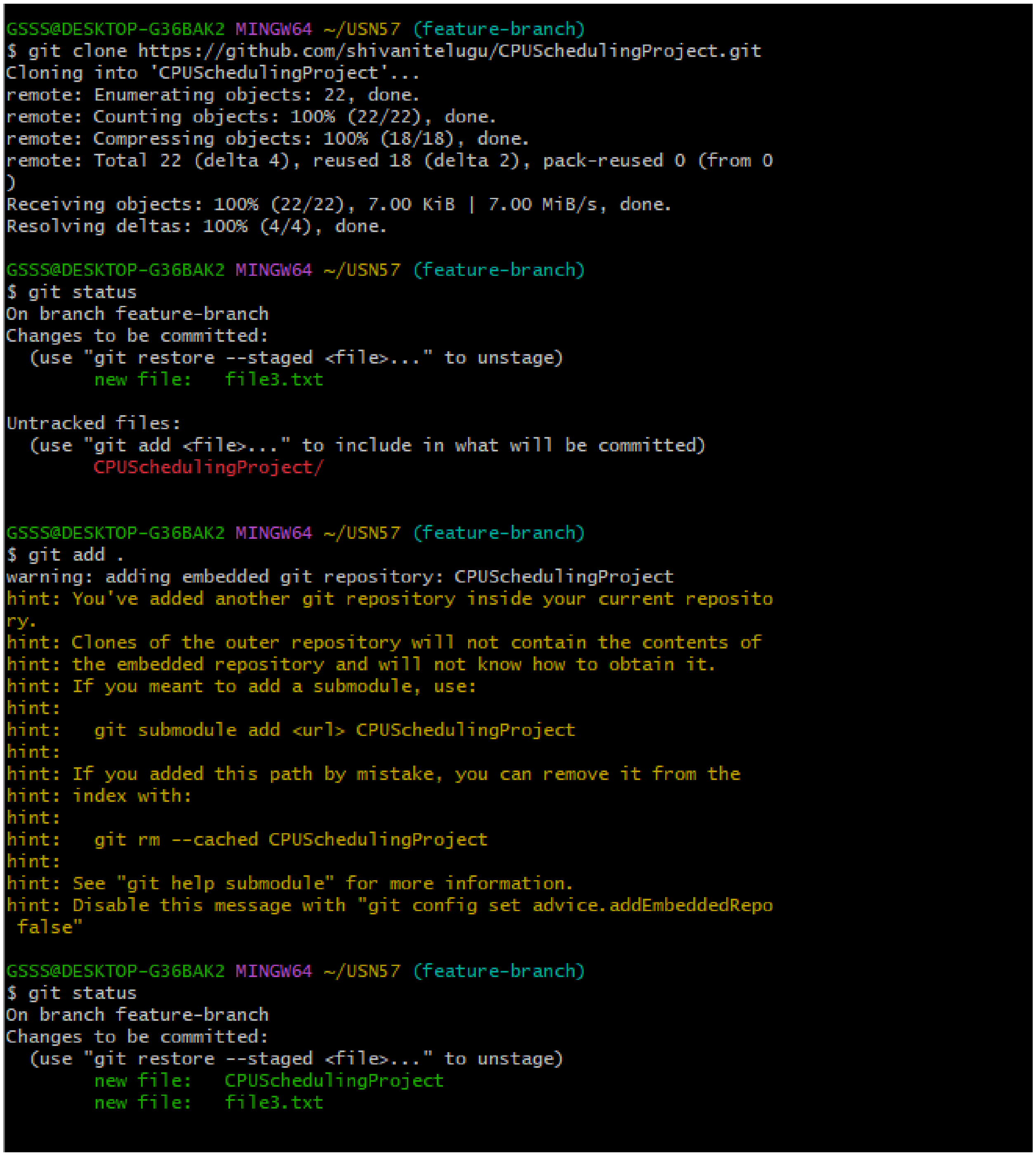
Step 1: Copy the remote repository URL

Step 2: Open terminal / Git Bash

Step 3: Clone the repository

```
git clone https://github.com/username/repository-name.git
```

EXPECTED OUTPUT:



```
GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git clone https://github.com/shivanitelugu/CPUSchedulingProject.git
Cloning into 'CPUSchedulingProject'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 22 (delta 4), reused 18 (delta 2), pack-reused 0 (from 0
)
Receiving objects: 100% (22/22), 7.00 KiB | 7.00 MiB/s, done.
Resolving deltas: 100% (4/4), done.

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git status
On branch feature-branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file3.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        CPUSchedulingProject/


GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git add .
warning: adding embedded git repository: CPUSchedulingProject
hint: You've added another git repository inside your current reposito
ry.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> CPUSchedulingProject
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached CPUSchedulingProject
hint:
hint: See "git help submodule" for more information.
hint: Disable this message with "git config set advice.addEmbeddedRepo
 false"

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git status
On branch feature-branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   CPUSchedulingProject
        new file:   file3.txt
```

Experiment 5:

Collaboration and Remote Repositories:

**Fetch the latest changes from a remote repository and rebase your local branch
onto the updated remote branch.**

Step 1: Check current branch

```
git branch
```

Step 2: Fetch latest changes from remote

```
git fetch origin
```

Step 3: Rebase local branch onto updated remote branch

```
git rebase origin/master
```

(Use main if your repo uses main branch)

```
git rebase origin/main
```

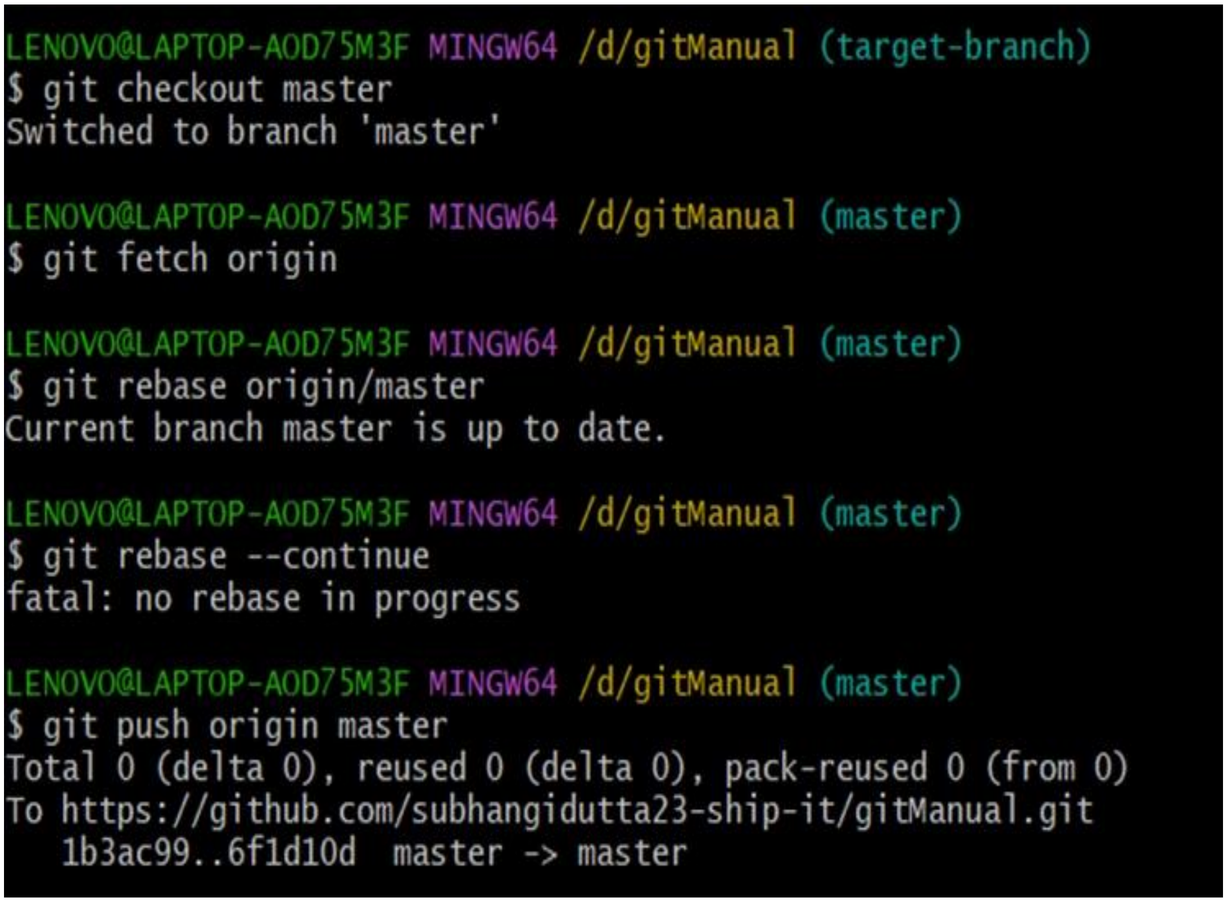
Step 4: Resolve conflicts (if any)

```
git status
```

Fix conflicts → then:

```
git add <file-name>
git rebase –continue

git push origin <branch-name>
```

EXPECTED OUTPUT:



```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (target-branch)
$ git checkout master
Switched to branch 'master'

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git fetch origin

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git rebase origin/master
Current branch master is up to date.

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git rebase --continue
fatal: no rebase in progress

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git push origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/subhangidutta23-ship-it/gitManual.git
   1b3ac99..6f1d10d  master -> master
```

Experiment 6:

Collaboration and Remote Repositories:

Write the command to merge "feature-branch" into "master" while providing a custom commit message for the merg

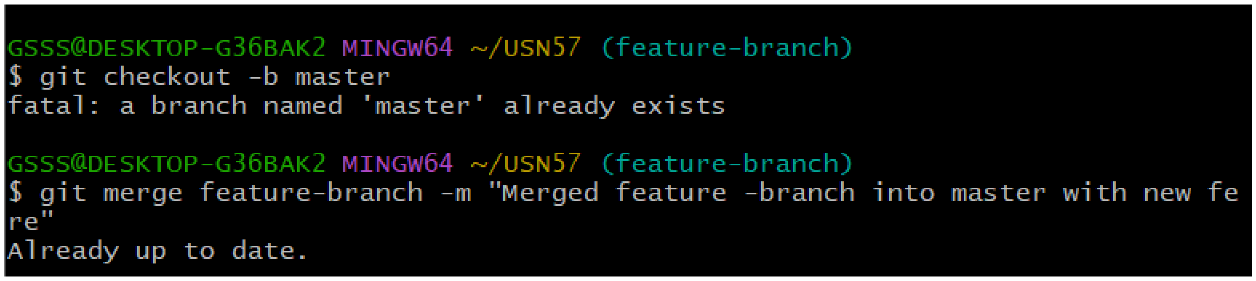
Solution:

To merge the "feature-branch" into "master" in Git while providing a custom commit message
for the merge, you can use the following command:

```
$ git checkout master
$ git merge feature-branch -m "Your custom commit message here"
```

Replace "Your custom commit message here" with a meaningful and descriptive commit message for the merge. This message will be associated with the merge commit that is created when you merge "feature-branch" into "master."

EXPECTED OUTPUT:



```
GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git checkout -b master
fatal: a branch named 'master' already exists

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git merge feature-branch -m "Merged feature -branch into master with new fe
re"
Already up to date.
```

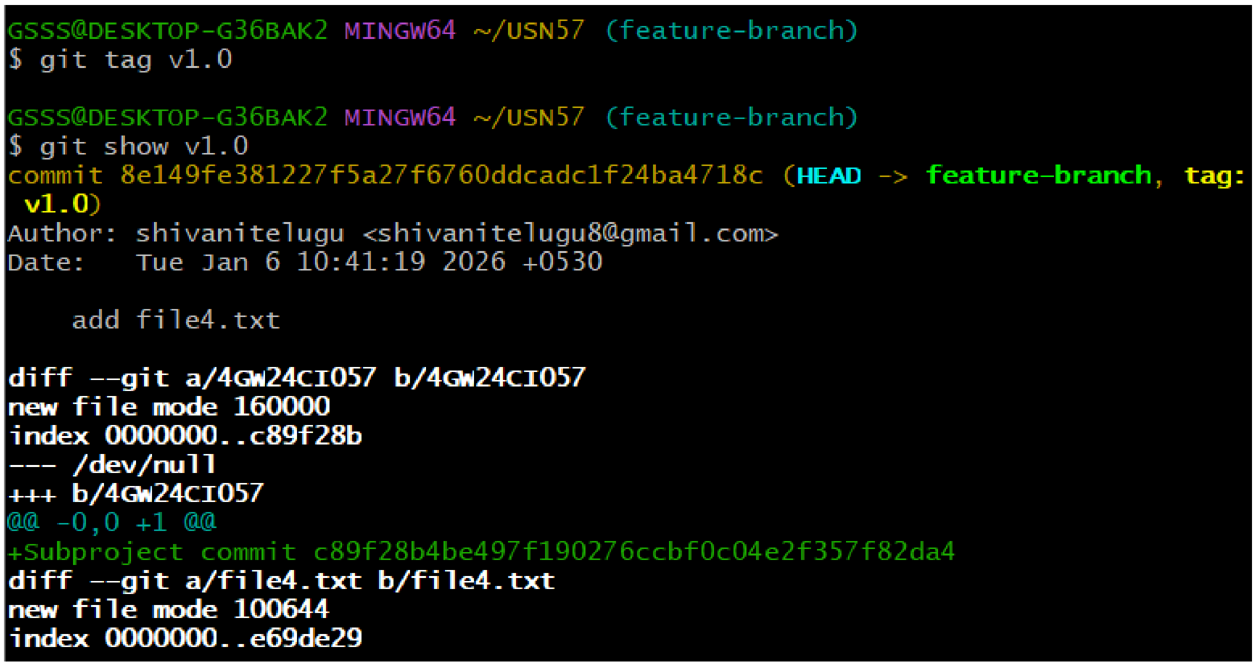
Experiment 7:

Git Tags and Releases:

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

Create a lightweight tag named v1.0 for the latest commit
git tag v1.0

EXPECTED OUTPUT:



```
GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git tag v1.0

GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git show v1.0
commit 8e149fe381227f5a27f6760ddcadc1f24ba4718c (HEAD -> feature-branch, tag:
 v1.0)
Author: shivanitelugu <shivanitelugu8@gmail.com>
Date:   Tue Jan 6 10:41:19 2026 +0530

    add file4.txt

diff --git a/4GW24CI057 b/4GW24CI057
new file mode 160000
index 0000000..c89f28b
--- /dev/null
+++ b/4GW24CI057
@@ -0,0 +1 @@
+Subproject commit c89f28b4be497f190276ccbf0c04e2f357f82da4
diff --git a/file4.txt b/file4.txt
new file mode 100644
index 0000000..e69de29
```

Experiment 8:

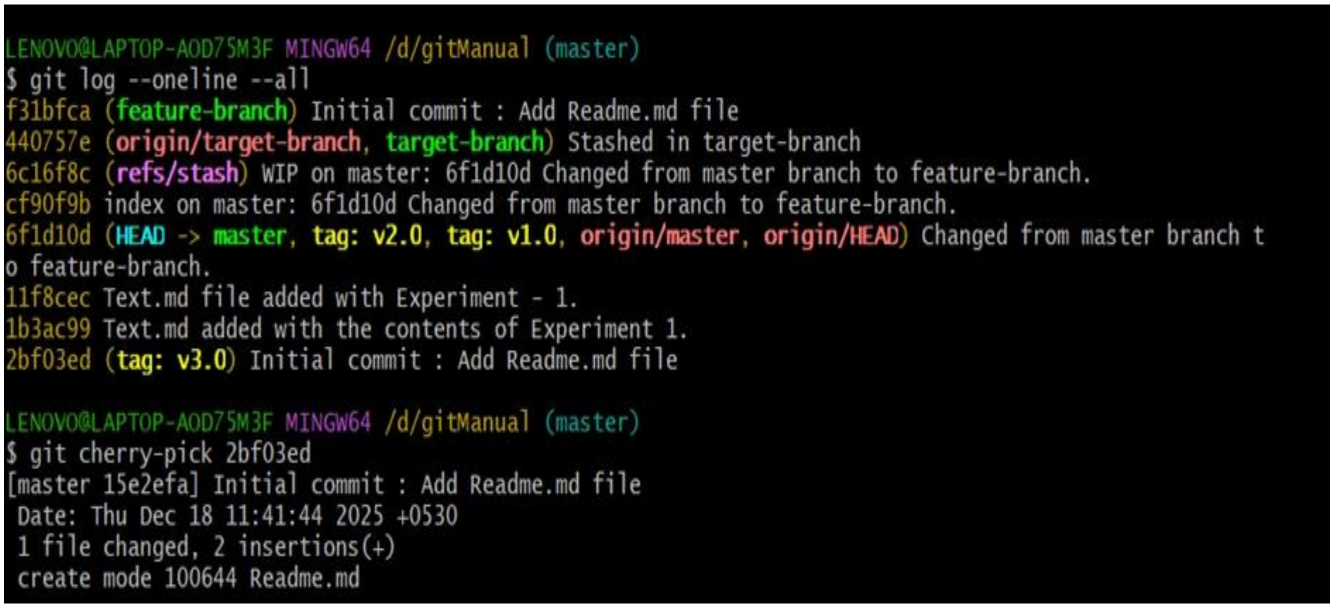
Advanced Git Operations:

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

Command to Cherry-pick a Range of Commits

git cherry-pick <start-commit>^..<end-commit>

EXPECTED OUTPUT:



```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git log --oneline --all
f31bfca (feature-branch) Initial commit : Add Readme.md file
440757e (origin/target-branch, target-branch) Stashed in target-branch
6c16f8c (refs/stash) WIP on master: 6f1d10d Changed from master branch to feature-branch.
cf90f9b index on master: 6f1d10d Changed from master branch to feature-branch.
6f1d10d (HEAD -> master, tag: v2.0, tag: v1.0, origin/master, origin/HEAD) Changed from master branch t
o feature-branch.
11f8cec Text.md file added with Experiment - 1.
1b3ac99 Text.md added with the contents of Experiment 1.
2bf03ed (tag: v3.0) Initial commit : Add Readme.md file

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git cherry-pick 2bf03ed
[master 15e2efa] Initial commit : Add Readme.md file
 Date: Thu Dec 18 11:41:44 2025 +0530
 1 file changed, 2 insertions(+)
 create mode 100644 Readme.md
```

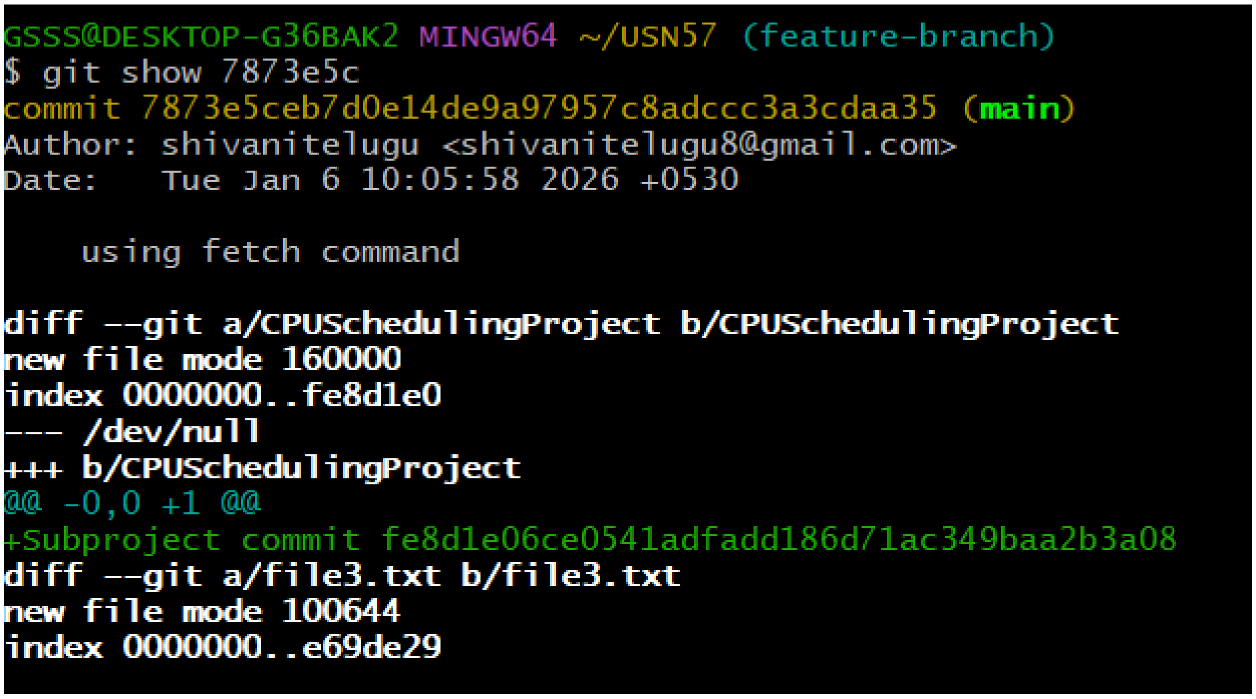
Experiment 9:

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?

Command to View Details of a Specific Commit
git show <commit-id>

EXPECTED OUTPUT:



```
GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git show 7873e5c
commit 7873e5ceb7d0e14de9a97957c8adccc3a3cdaa35 (main)
Author: shivanitelugu <shivanitelugu8@gmail.com>
Date:   Tue Jan 6 10:05:58 2026 +0530

    using fetch command

diff --git a/CPUSchedulingProject b/CPUSchedulingProject
new file mode 160000
index 0000000..fe8d1e0
--- /dev/null
+++ b/CPUSchedulingProject
@@ -0,0 +1 @@
+Subproject commit fe8d1e06ce0541adfadd186d71ac349baa2b3a08
diff --git a/file3.txt b/file3.txt
new file mode 100644
index 0000000..e69de29
```

Experiment 10:

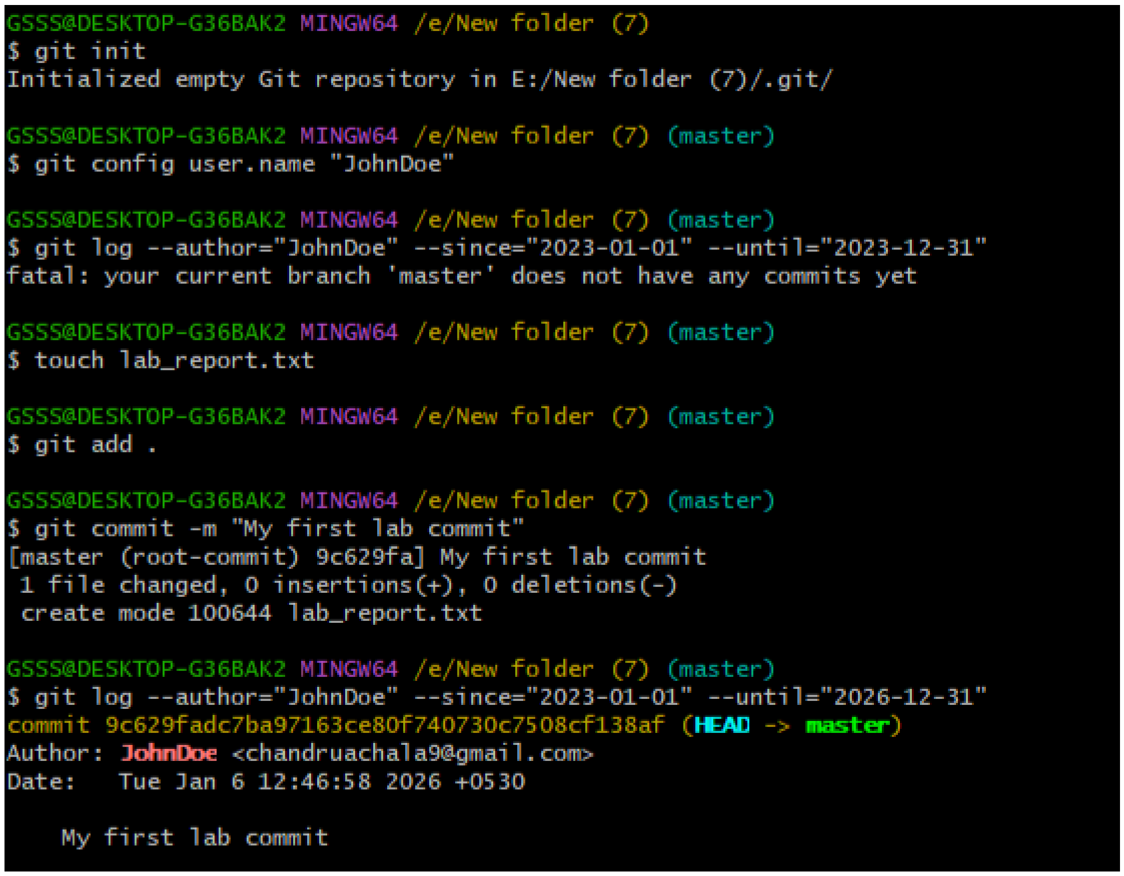
Analysing and Changing Git History

Write the command to list all commits made by the author "JohnDoe" between "2023-01-01" and "2023-12-31."

Command to List Commits by a Specific Author Within a Date Range

git log --author="JohnDoe" --since="2023-01-01" --until="2023-12-31"

EXPECTED OUTPUT:



```
GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (7)
$ git init
Initialized empty Git repository in E:/New folder (7)/.git/

GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (7) (master)
$ git config user.name "JohnDoe"

GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (7) (master)
$ git log --author="JohnDoe" --since="2023-01-01" --until="2023-12-31"
fatal: your current branch 'master' does not have any commits yet

GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (7) (master)
$ touch lab_report.txt

GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (7) (master)
$ git add .

GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (7) (master)
$ git commit -m "My first lab commit"
[master (root-commit) 9c629fa] My first lab commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 lab_report.txt

GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (7) (master)
$ git log --author="JohnDoe" --since="2023-01-01" --until="2026-12-31"
commit 9c629fadc7ba97163ce80f740730c7508cf138af (HEAD -> master)
Author: JohnDoe <chandruachala9@gmail.com>
Date:   Tue Jan 6 12:46:58 2026 +0530

    My first lab commit
```

Experiment 11:

Analysing and Changing Git History

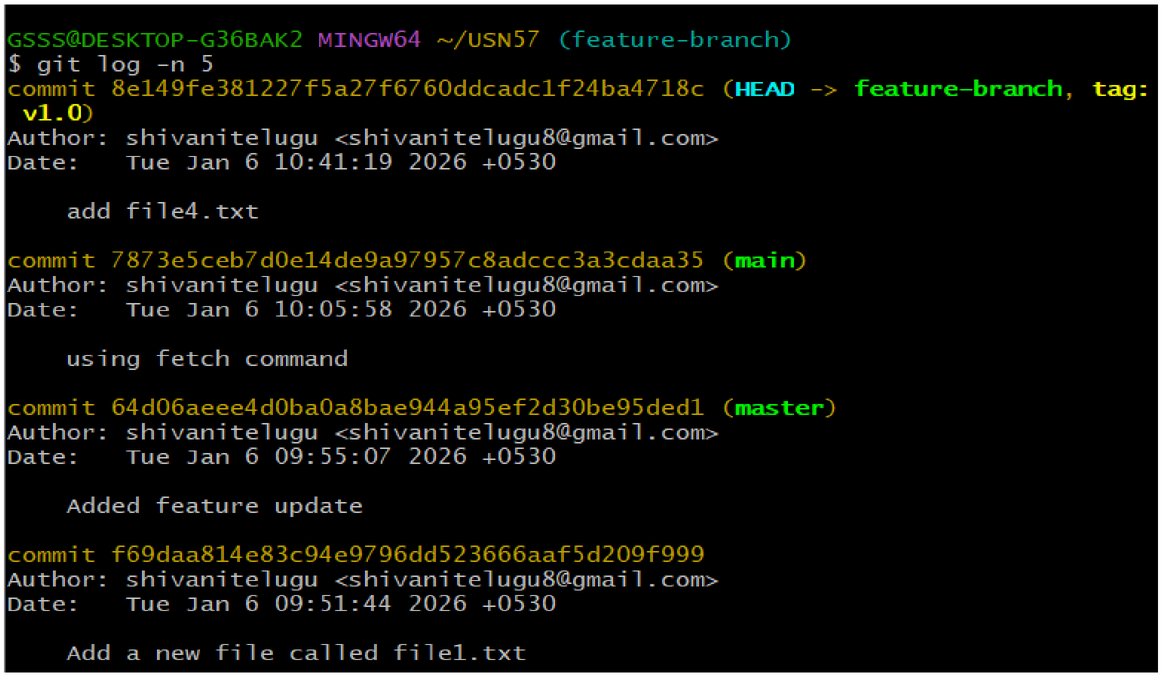
Write the command to display the last five commits in the repository's history.

To display the last five commits in a Git repository's history, you can use the git log command with the -n option, which limits the number of displayed commits. Here's the command:

```
$ git log -n 5
```

This command will show the last five commits in the repository's history. You can adjust the number after -n to display a different number of commits if needed

EXPECTED OUTPUT:



```
GSSS@DESKTOP-G36BAK2 MINGW64 ~/USN57 (feature-branch)
$ git log -n 5
commit 8e149fe381227f5a27f6760ddcadc1f24ba4718c (HEAD -> feature-branch, tag:
 v1.0)
Author: shivanitelugu <shivanitelugu8@gmail.com>
Date:   Tue Jan 6 10:41:19 2026 +0530

    add file4.txt

commit 7873e5ceb7d0e14de9a97957c8adccc3a3cdaa35 (main)
Author: shivanitelugu <shivanitelugu8@gmail.com>
Date:   Tue Jan 6 10:05:58 2026 +0530

    using fetch command

commit 64d06aeee4d0ba0a8bae944a95ef2d30be95ded1 (master)
Author: shivanitelugu <shivanitelugu8@gmail.com>
Date:   Tue Jan 6 09:55:07 2026 +0530

    Added feature update

commit f69daa814e83c94e9796dd523666aaf5d209f999
Author: shivanitelugu <shivanitelugu8@gmail.com>
Date:   Tue Jan 6 09:51:44 2026 +0530

    Add a new file called file1.txt
```

Experiment 12:

Analysing and Changing Git History

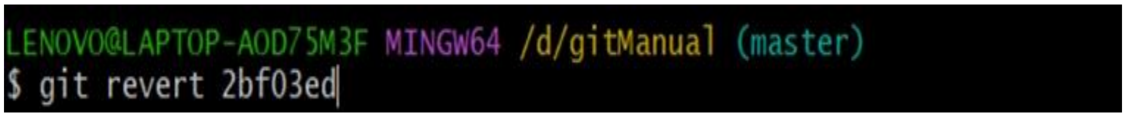
Write the command to undo the changes introduced by the commit with the ID "abc123".

To undo the changes introduced by a specific commit with the ID "abc123" in Git, you can
use the git revert command. The git revert command creates a new commit that undoes the
changes made by the specified commit, effectively "reverting" the commit. Here's the command:

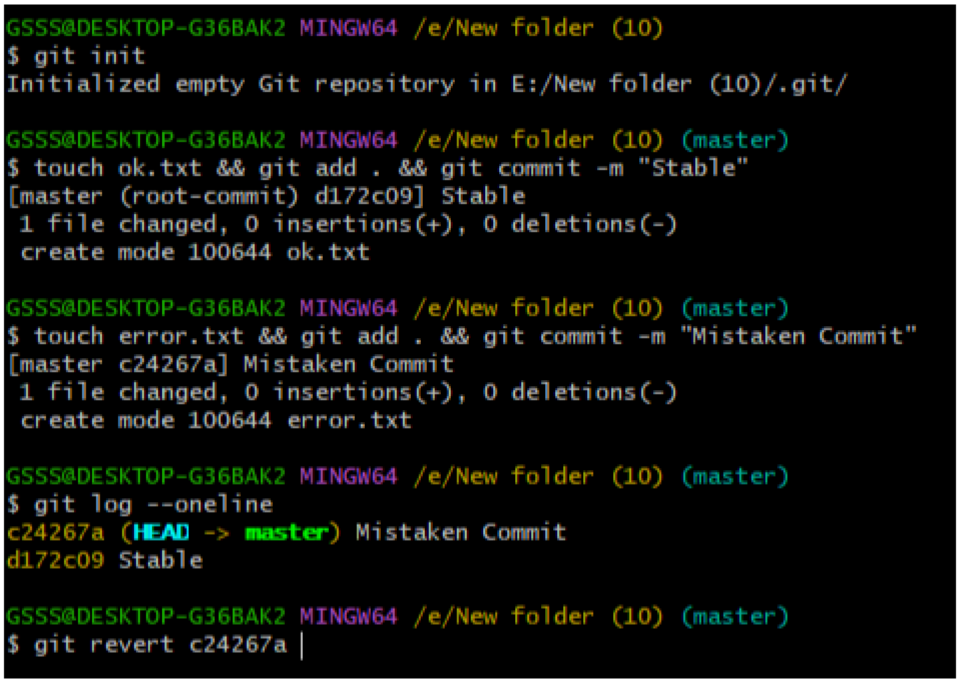
```
$ git revert abc123
```

Replace "abc123" with the actual commit ID that you want to revert. After running this
command, Git will create a new commit that negates the changes introduced by the specified
commit. This is a safe way to undo changes in Git because it preserves the commit history and
creates a new commit to record the reversal of the changes.

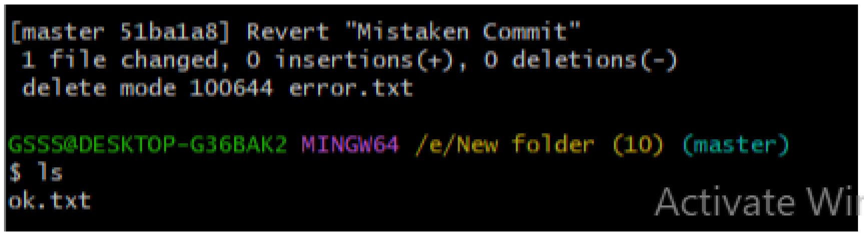
EXPECTED OUTPUT:



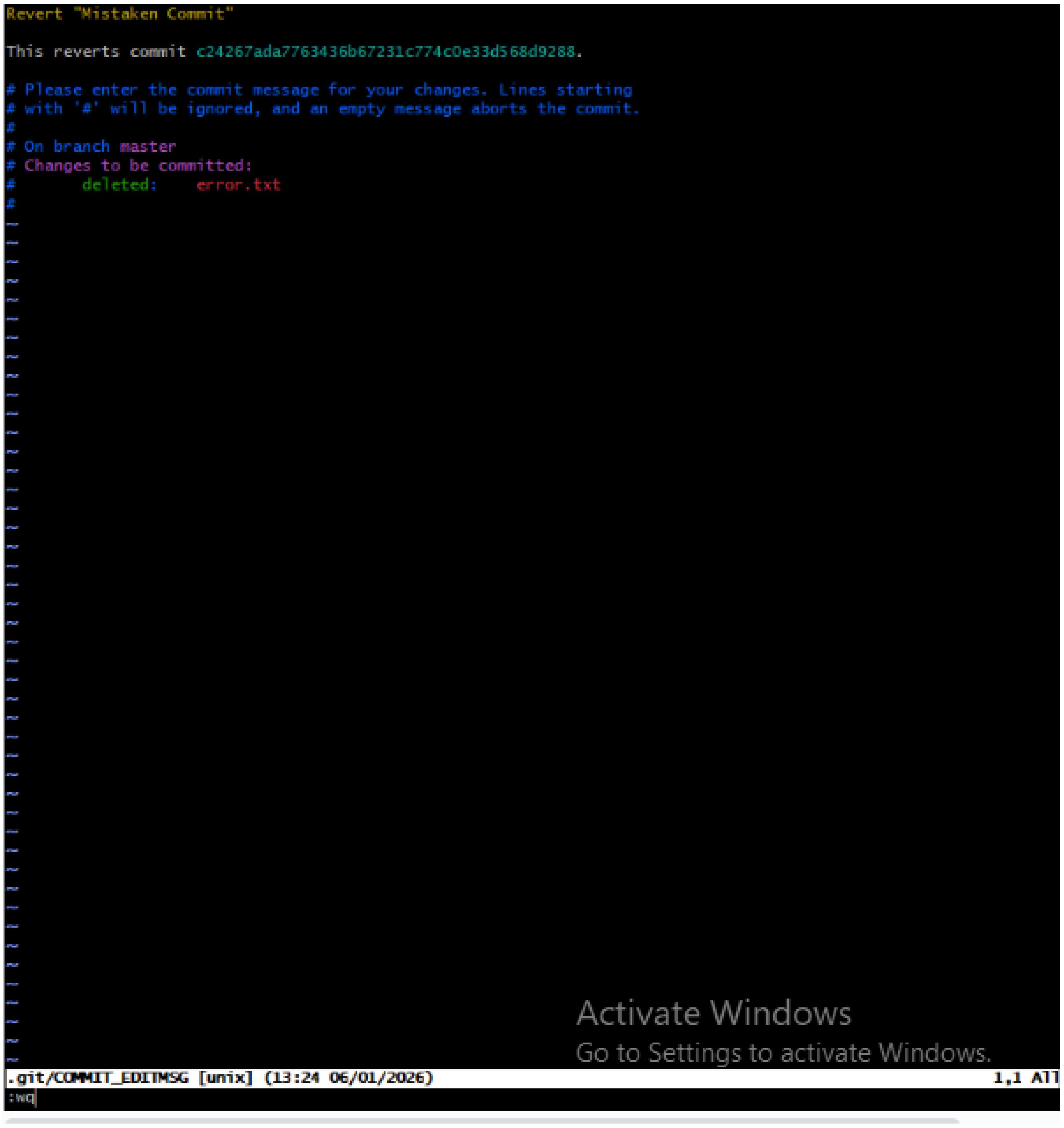
```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git revert 2bf03ed
```



```
GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (10)
$ git init
Initialized empty Git repository in E:/New folder (10)/.git/

GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (10) (master)
$ touch ok.txt && git add . && git commit -m "Stable"
[master (root-commit) d172c09] Stable
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ok.txt

GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (10) (master)
$ touch error.txt && git add . && git commit -m "Mistaken Commit"
[master c24267a] Mistaken Commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 error.txt

GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (10) (master)
$ git log --oneline
c24267a (HEAD -> master) Mistaken Commit
d172c09 Stable

GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (10) (master)
$ git revert c24267a
```



```
[master 51ba1a8] Revert "Mistaken Commit"
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 error.txt

GSSS@DESKTOP-G36BAK2 MINGW64 /e/New folder (10) (master)
$ ls
ok.txt
```



```
Revert "Mistaken Commit"

This reverts commit c24267ada7763436b67231c774c0e33d568d9288.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#       deleted:    error.txt
#
~
.git/COMMIT_EDITMSG [unix] (13:24 06/01/2026)                    1,1 All
:wq
```

Activate Windows
Go to Settings to activate Windows.

Appendix

Repository Link

https://github.com/shivanitelugu/4GW24CI057.git