# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**"Jnana Sangama", Machhe, Belagavi, Karnataka-590018**


Lab Experiment Record

## Project Management with Git [BCSL358C]

*Submitted in partial fulfillment towards AEC of 3rd semester of*

**Bachelor of Engineering**
**in**
**Computer Science and Engineering**
**(Artificial Intelligence & Machine Learning)**

Submitted by

**SNEHAGANGA N S**
**4GW24CI048**

**DEPARTMENT OF CSE (Artificial Intelligence & Machine Learning)**

**GSSS INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN**

**(Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi & Govt. of Karnataka)**

**K.R.S ROAD, METAGALLI, MYSURU-570016, KARNATAKA**

**(Accredited by NAAC)**

**2025-2026**

## Contents

# Program 1: Setting Up and Basic Commands

**Step 1: Configure Git User Information**
git config --global user.name "SnehagangaNS"
git config --global user.email "snehaganganadumane@gmail.com"

- This command sets the user name and email for Git.

**Step 2: Initialize Git Repository**
git init

- This command creates a new Git repository in the current folder.

**Step 3: Create a File**
touch test.txt

- This command creates a new empty file named test.txt.

**Step 4: Check Status**
git status

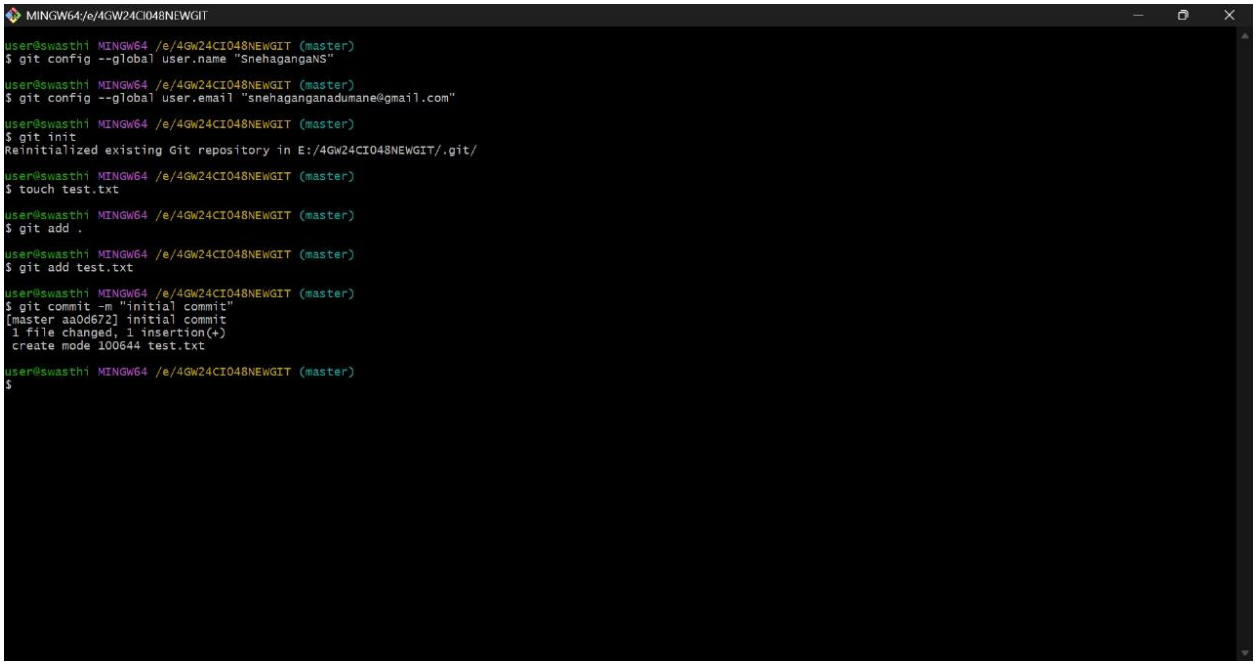
- This command shows the current status of the repository.

**Step 5: Add Files**
git add .

- This command adds all files to the staging area.

**Step 6: Commit the Changes**
git commit -m "Initial commit: added file1.txt"

- This command saves the changes to the repository.


MINGW64:/e/4GW24CI048NEWGIT
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git config --global user.name "SnehagangaNS"
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git config --global user.email "snehaganganadumane@gmail.com"
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git init
Reinitialized existing Git repository in E:/4GW24CI048NEWGIT/.git/
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ touch test.txt
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git add .
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git add test.txt
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git commit -m "initial commit"
[master aa0d672] initial commit
1 file changed, 1 insertion(+)
create mode 100644 test.txt
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$

# Program 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: Configure Git Globally**
git config --global user.name "SnehagangaNS"
git config --global user.email "snehaganganaumane@gmail.com"

- This command sets the user name and email for Git.

**Step 2: Initialize a New Git Repository**
git init

- This command creates a new Git repository in the current directory.

**Step 3: Create a File and Make the First Commit**
touch test.txt
git add .
git commit -m "Initial commit"

- A file named test.txt is created in the main directory.
- The file is added.
- The first commit is made, which is required before creating branches.

**Step 4: Create a New Branch**
git branch feature-branch

- This command creates a new branch called feature-branch.

**Step 5: Switch to the Feature Branch**
git checkout feature-branch

- This command switches from the master branch to the feature branch.

**Step 6: Make Changes in Feature Branch**
git add .
git commit -m "Added changes in feature-branch"

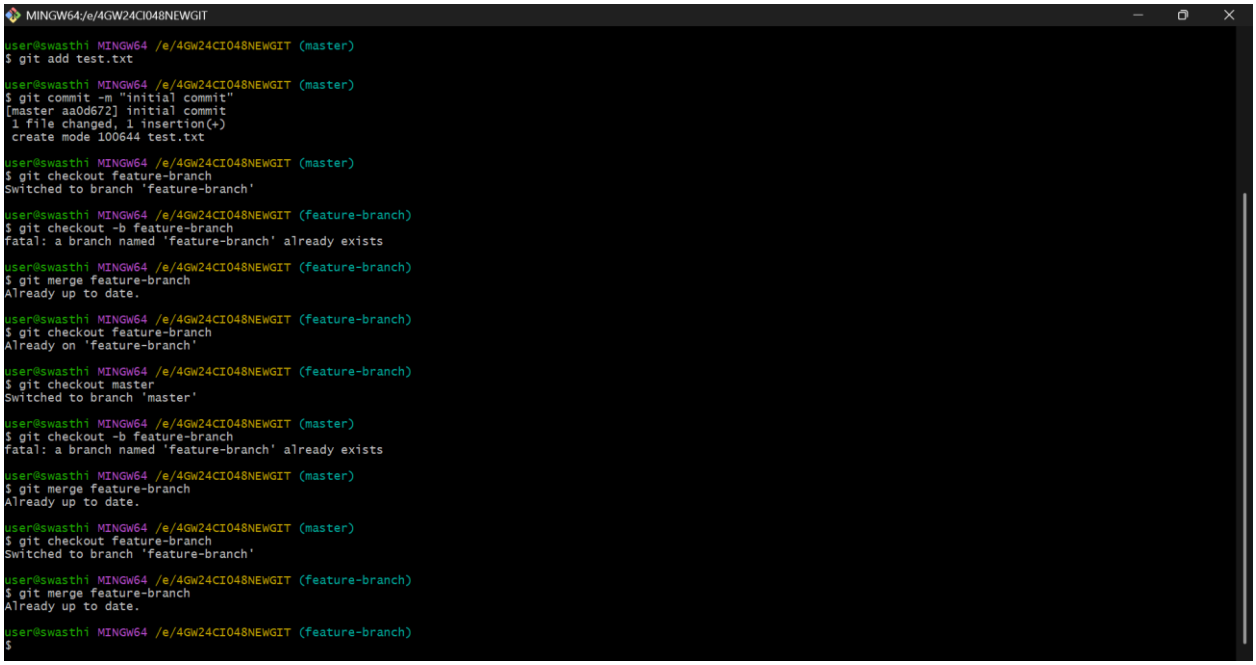
- Changes are made in the feature branch.
- The updated files are added and committed.

**Step 7: Switch Back to Master Branch**
git checkout master

- This command switches back to the master branch.

**Step 8: Merge Feature Branch into Master**
git merge feature-branch

- This command merges the changes from feature-branch into master.


MINGW64:/e/4GW24CI048NEWGIT
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git add test.txt
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git commit -m "initial commit"
[master aa0d672] initial commit
1 file changed, 1 insertion(+)
create mode 100644 test.txt
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git checkout feature-branch
Switched to branch 'feature-branch'
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git checkout -b feature-branch
fatal: a branch named 'feature-branch' already exists
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git merge feature-branch
Already up to date.
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git checkout feature-branch
Already on 'feature-branch'
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git checkout master
Switched to branch 'master'
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git checkout -b feature-branch
fatal: a branch named 'feature-branch' already exists
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git merge feature-branch
Already up to date.
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git checkout feature-branch
Switched to branch 'feature-branch'
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git merge feature-branch
Already up to date.
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$

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

**Step 1: Configure Git User Information**

git config --global user.name "SnehagangaNS"
git config --global user.email "snehaganganadumane@gmail.com"

- Sets the username and email for Git commits.

**Step 2: Initialize a Git Repository**

git init

- Creates a new Git repository in the current directory.

**Step 3: Create a File and Make Initial Commit**

touch test.txt
git add .
git commit -m "Initial commit"

- Creates a file named test.txt.
- Adds the file and commits it to the repository.
- This first commit is required before using branches.

**Step 4: Create and Switch to a Feature Branch**

git branch feature-branch
git checkout feature-branch

- Creates a new branch called feature-branch.
- Switches from the master branch to the feature branch.

**Step 6: Switch Back to Master Branch**

git checkout master

- Returns to the master branch.
- The file test.txt still exists in this branch.

**Step 7: Stash the Current Changes**

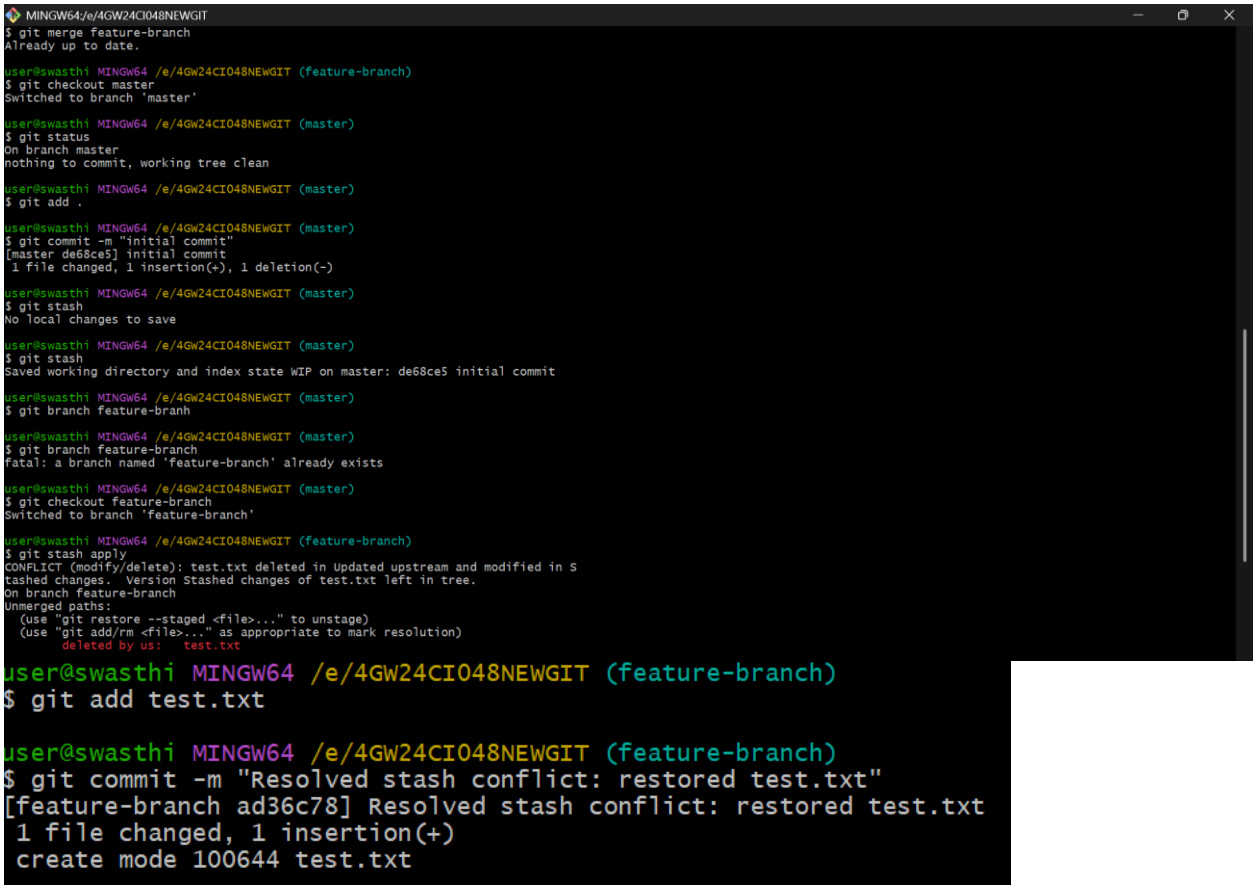
git stash

- Temporarily saves uncommitted changes.

**Step 8: Switch to Feature Branch and Apply Stash**

git checkout feature-branch
git stash apply

- Switches back to the feature branch.
- Applies the stashed changes to this branch.


MINGW64:/e/4GW24CI048NEWGIT
$ git merge feature-branch
Already up to date.
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git checkout master
Switched to branch 'master'
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git status
On branch master
nothing to commit, working tree clean
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git add .
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git commit -m "initial commit"
[master de68ce5] initial commit
1 file changed, 1 insertion(+), 1 deletion(-)
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git stash
No local changes to save
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git stash
Saved working directory and index state WIP on master: de68ce5 initial commit
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git branch feature-branh
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git branch feature-branch
fatal: a branch named 'feature-branch' already exists
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git checkout feature-branch
Switched to branch 'feature-branch'
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git stash apply
CONFLICT (modify/delete): test.txt deleted in Updated upstream and modified in S
tashed changes. Version Stashed changes of test.txt left in tree.
On branch feature-branch
Unmerged paths:
(use "git restore --staged <file>..." to unstage)
(use "git add/rm <file>..." as appropriate to mark resolution)
deleted by us: test.txt
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git add test.txt
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git commit -m "Resolved stash conflict: restored test.txt"
[feature-branch ad36c78] Resolved stash conflict: restored test.txt
1 file changed, 1 insertion(+)
create mode 100644 test.txt

## Program 4: Collaboration and Remote Repositories Clone a remote Git repository to your local machine.

**Step 1: Configure Git User Details**

git config --global user.name "SnehagangaNS"
git config --global user.email "snehaganganadumane@gmail.com"

- Sets the user name and email address for Git.

**Step 2: Clone a Remote Repository**

git clone https://github.com/varsha-sureshh/SignLingo_TechSprint_Hackathon.git

- This command creates a local copy of the remote repository from GitHub.

```
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git clone https://github.com/varsha-sureshh/SignLingo_TechSprint_Hackathon.git
Cloning into 'SignLingo_TechSprint_Hackathon'...
remote: Enumerating objects: 70, done.
remote: Counting objects: 100% (70/70), done.
remote: Compressing objects: 100% (54/54), done.
remote: Total 70 (delta 20), reused 66 (delta 16), pack-reused 0 (from 0)
Receiving objects: 100% (70/70), 24.14 MiB | 4.39 MiB/s, done.
Resolving deltas: 100% (20/20), done.
```

## Program 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: Switch to Feature Branch**

git checkout feature-branch

- This command switches the current working branch to feature-branch.

**Step 2: Fetch Latest Changes from Remote Repository**

git fetch origin

- This command downloads the latest changes from the remote repository.

**Step 3: Rebase Feature Branch onto Remote Master**

git rebase origin/master

- This command reapplies the commits from feature-branch on top of origin/master.

```
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git branch
* feature-branch
  feature-branh
  master

user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git fetch origin
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 21 (delta 2), reused 16 (delta 2), pack-reused 0 (from 0)
Unpacking objects: 100% (21/21), 3.60 KiB | 61.00 KiB/s, done.
From https://github.com/SnehaganganS/4GW24CI048GIT
 * [new branch]      main       -> origin/main

user@swasthi MINGW64 /e/4GW24CI048NEWGIT (feature-branch)
$ git rebase origin origin/master
Successfully rebased and updated detached HEAD.
```

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

**Step 1: Switch to Master Branch**

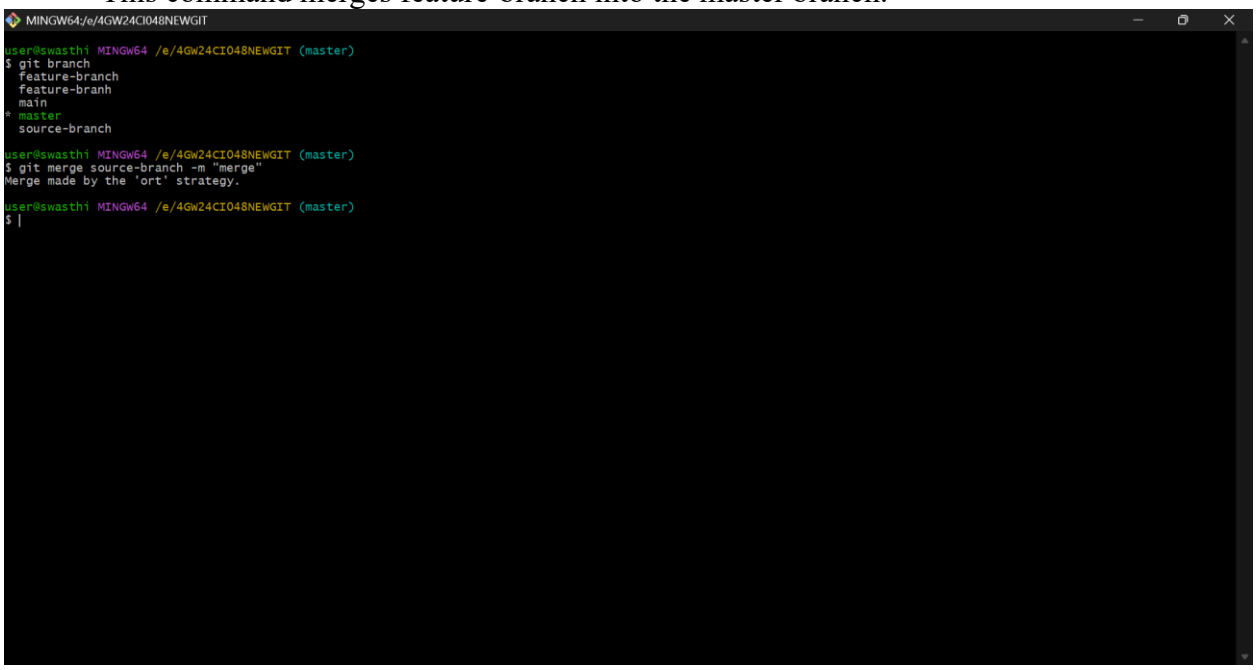
git checkout master

- This command switches the current branch to master.

**Step 2: Merge Feature Branch into Master**

git merge source-branch -m "Merged source-branch into master"

- This command merges feature-branch into the master branch.

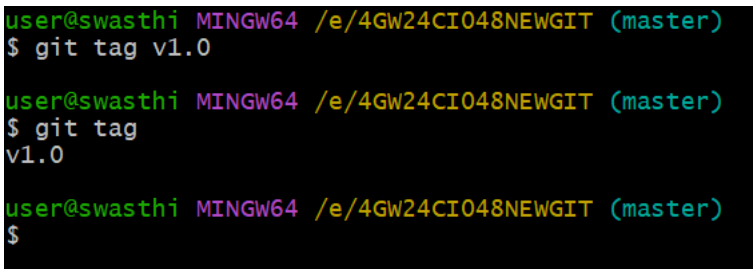


## Program 7: Git Tags and Releases Write the command to create a lightweight Git tag named "v1.0" for a commit in your local repository.

**Step: Create a Tag**

git tag v1.0

- This command creates a tag named v1.0.
- A tag is used to mark a specific point in the project history.



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

**Step 1: Create and Switch to Source Branch**

git checkout -b source-branch

- This command creates a new branch named source-branch.

**Step 2: Make Commits in Source Branch**

git add .
git commit -m "First commit in source-branch"
git add .
git commit -m "Second commit in source-branch"

- Files are edited manually before each commit.
- Each commit saves a separate set of changes.
- Multiple commits are created to demonstrate cherry-picking.

**Step 3: Switch Back to Target (Master) Branch**

git checkout master

- This command switches back to the master branch.

**Step 4: View Commits in Source Branch**

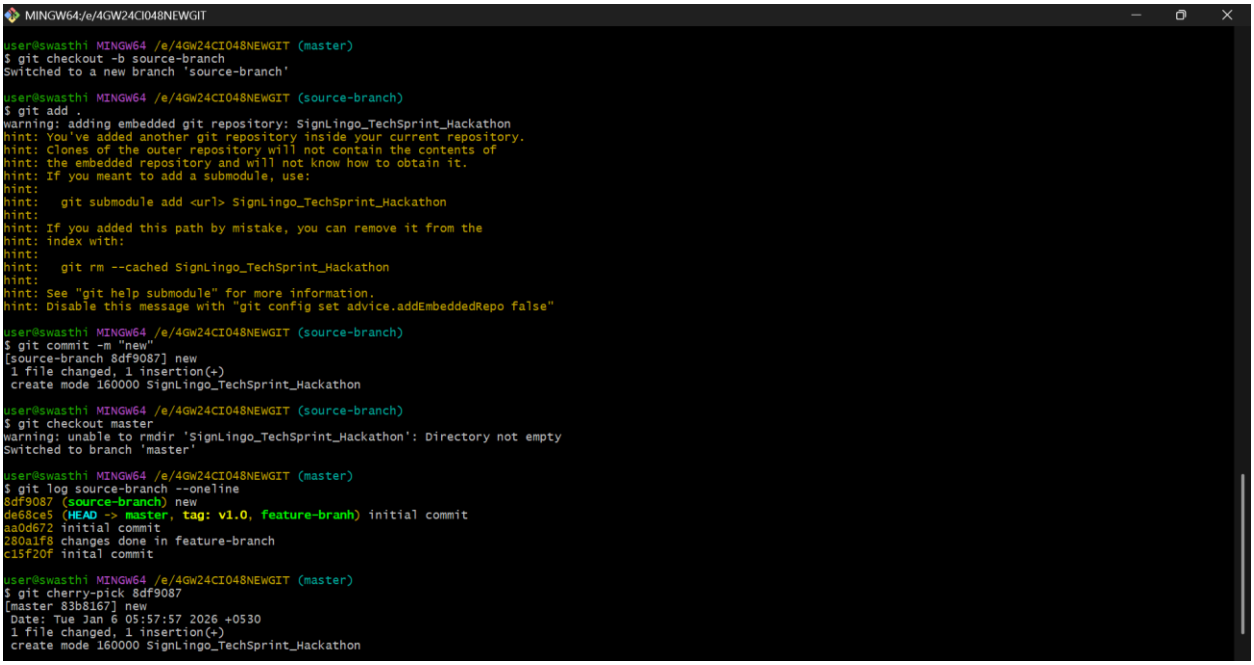
git log source-branch --oneline

- Displays the list of commits made in source-branch.
- Shows commit IDs in a short format.
- These commit IDs are needed for cherry-picking.

**Step 5: Cherry-Pick a Range of Commits**

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

- This command copies a specific range of commits from source-branch.
- The selected commits are applied to the master branch.
- Only chosen commits are merged, not the entire branch.


MINGW64:/e/4GW24CI048NEWGIT
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git checkout -b source-branch
Switched to a new branch 'source-branch'
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (source-branch)
$ git add .
warning: adding embedded git repository: SignLingo_TechSprint_Hackathon
hint: You've added another git repository inside your current repository.
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> SignLingo_TechSprint_Hackathon
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint: git rm --cached SignLingo_TechSprint_Hackathon
hint:
hint: See "git help submodule" for more information.
hint: Disable this message with "git config set advice.addEmbeddedRepo false"
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (source-branch)
$ git commit -m "new"
[source-branch 8df9087] new
1 file changed, 1 insertion(+)
create mode 160000 SignLingo_TechSprint_Hackathon
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (source-branch)
$ git checkout master
warning: unable to rmdir 'SignLingo_TechSprint_Hackathon': Directory not empty
Switched to branch 'master'
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git log source-branch --oneline
8df9087 (source-branch) new
de68ce5 (HEAD -> master, tag: v1.0, feature-branh) initial commit
aa0d672 initial commit
280a1f8 changes done in feature-branch
c15f20f inital commit
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git cherry-pick 8df9087
[master 83b8167] new
Date: Tue Jan 6 05:57:57 2026 +0530
1 file changed, 1 insertion(+)
create mode 160000 SignLingo_TechSprint_Hackathon

## Program 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.

### Step: Display Commit Information

git show <commit-id>

- This command displays detailed information about a specific commit.
- It shows the commit author, date, commit message, and the changes made.

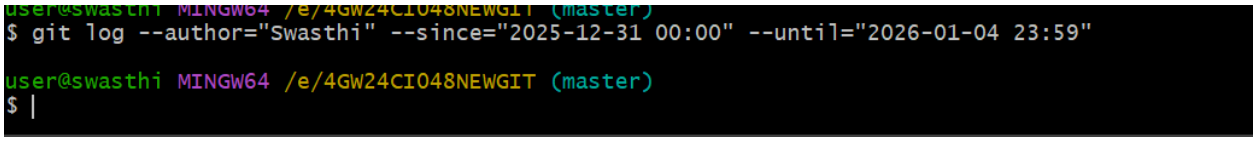
```
$ git show de68ce5
commit de68ce5e62c8454c752384818ea8660570d2ebc6 (tag: v1.0, feature-branh)
Author: SnehagangaNS <snehaganganadumane@gmail.com>
Date:   Tue Jan 6 05:11:46 2026 +0530

    initial commit

diff --git a/test.txt b/test.txt
index fabfbb2..9394436 100644
--- a/test.txt
+++ b/test.txt
@@ -1 +1 @@
-hi hello
\ No newline at end of file
+hi hello namsate
\ No newline at end of file
```

## Program 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."

**Explanation**

git log --author="Swasthi" --since="2025-01-01" --until="2026-12-31"

- git log shows the commit history of the repository.
- --author="Swasthi" filters and displays only the commits made by the author **Swasthi**.
- --since="2025-01-01" shows commits starting from January 1, 2025.
- --until="2026-12-31" limits the results up to December 31, 2026.
- Only commits that match both the author name and the date range are displayed.



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

**Explanation**
git log -n 5

- git log displays the commit history of the repository.
- -n 5 limits the output to the last 5 commits only.
- The commits are shown in reverse chronological order .
- Each entry includes the commit ID, author, date, and commit message.

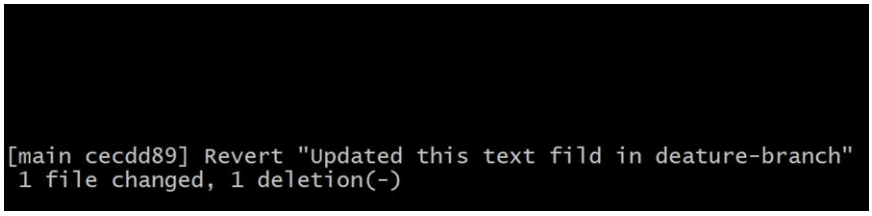
```
user@swasthi MINGW64 /e/4GW24CI048NEWGIT (master)
$ git log -n 5
commit 83b81677768d4093cc193a52574fe36a425e6838 (HEAD -> master)
Author: SnehaganganS <snehaganganadumane@gmail.com>
Date:   Tue Jan 6 05:57:57 2026 +0530

    new

commit de68ce5e62c8454c752384818ea8660570d2ebc6 (tag: v1.0, feature-branh)
Author: SnehaganganS <snehaganganadumane@gmail.com>
Date:   Tue Jan 6 05:11:46 2026 +0530

    initial commit

commit aa0d67202404d45167220234cd46cd6640c1ed6b
Author: SnehaganganS <snehaganganadumane@gmail.com>
Date:   Tue Jan 6 04:43:21 2026 +0530

    initial commit

commit 280a1f887930d43dd50ead0ce94d13fa36bf9552
Author: Your Name <snehaganganadumane@gmail.com>
Date:   Mon Jan 5 19:07:44 2026 +0530

    changes done in feature-branch

commit c15f20f2856316529d907598db20894aa31cf95e
Author: Your Name <snehaganganadumane@gmail.com>
Date:   Mon Jan 5 19:06:18 2026 +0530

    inital commit
```

## Program 12: Analyzing and Changing Git History Write the command to undo the changes introduced by the commit with the ID "abc123".

**Explanation**

git revert abc123

- git revert is used to undo the changes made by a specific commit.
- abc123 represents the commit ID (hash) that needs to be reverted.
- Instead of deleting history, Git creates a new commit that reverses the changes of the selected commit.
- This method is safe and recommended for shared repositories.



# APPENDIX

https://github.com/SnehagangaNS/4GW24CI048gitmanual.git