VISVESVARAYA TECHNOLOGICAL UNIVERSITY

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

Lab Experiment Record

Project Management with Git [BCS358C]

Submitted in partial fulfillment towards AEC of 3rd semester of

Bachelor of Engineering
in
Computer Science and Engineering

(Artificial Intelligence & Machine Learning)

Submitted by

SUBHANGI DUTTA

4GW24CI053

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

Index

A. Git Installation:

1. Installing Git on Windows: Using Git for Windows (Git Bash):

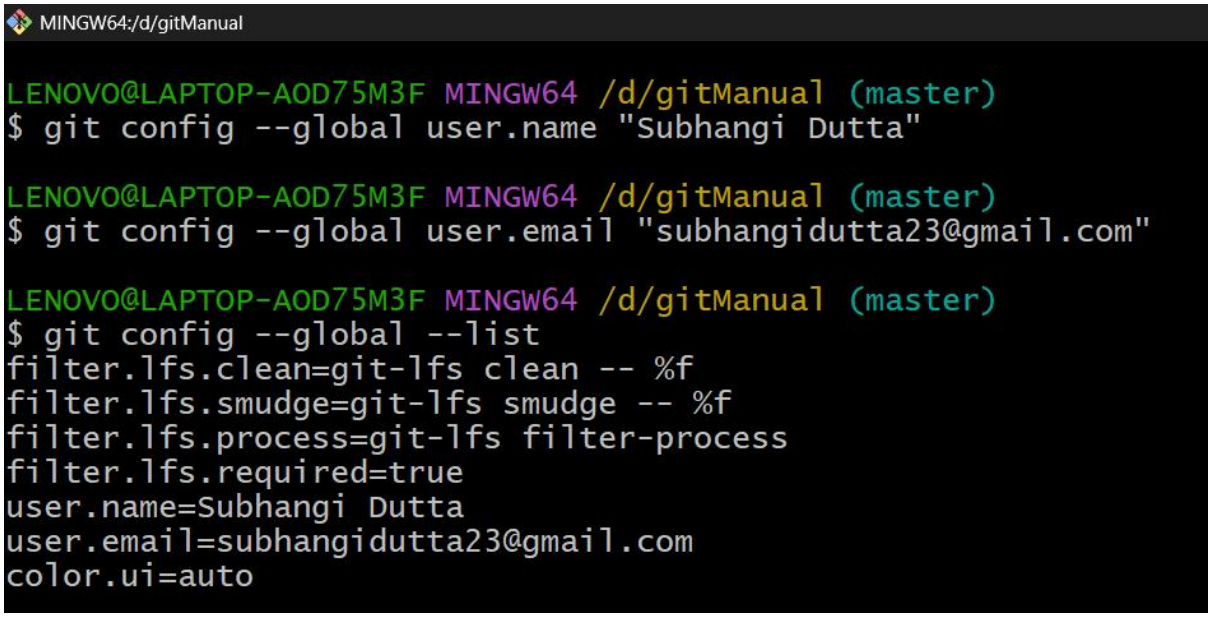
- Go to the official Git for Windows website: https://gitforwindows.org/
- Download the latest version of Git for Windows.
- Run the installer and follow the installation steps. You can choose the default settings for most options.

B. Git Configuration:

1. 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



C. Git Experiments:

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.

Solution:

To initialize a new Git repository in a directory, create a new file, add it to the staging area, and commit the changes with an appropriate commit message, follow these steps:

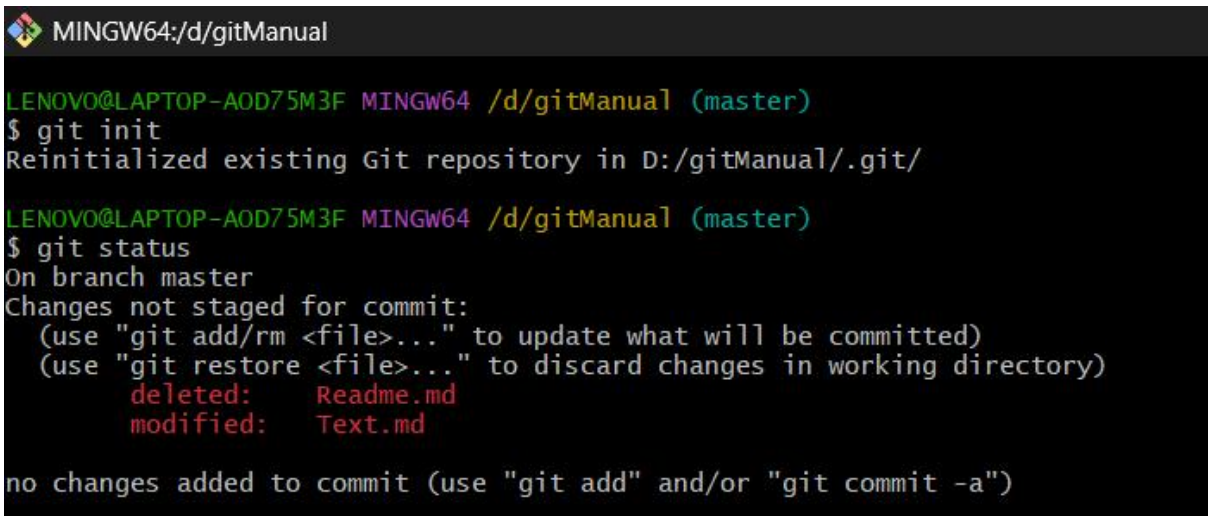
1. Open your terminal and navigate to the directory where you want to create the Git repository.

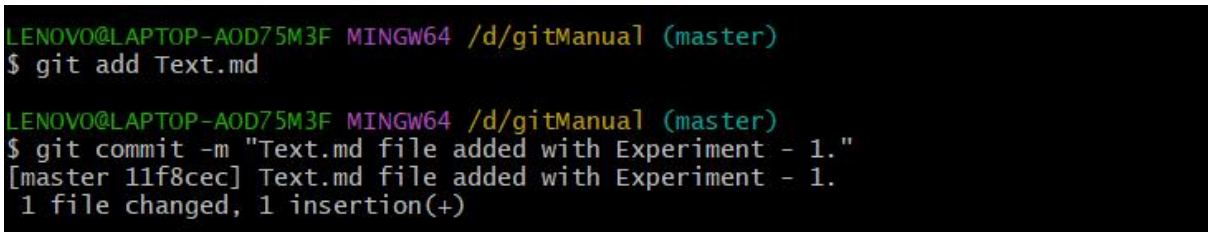
2. Initialize a new Git repository in that directory:
$ git init

3. Create a new file in the directory. For example, let's create a file named "my_file.txt." You can use any text editor or command-line tools to create the file.

4. Add the newly created file to the staging area. Replace "Text.md" with the actual name of your file:
$ git add Text.md
This command stages the file for the upcoming commit.

5. Commit the changes with an appropriate commit message. Replace "Your commit message here" with a meaningful description of your changes:
$ git commit -m "Your commit message here"
Your commit message should briefly describe the purpose or nature of the changes you made.
For example:
$ git commit -m "Add a new file called Text.md "
After these steps, your changes will be committed to the Git repository with the provided commit message. You now have a version of the repository with the new file and its history stored in Git.


MINGW64:/d/gitManual
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git init
Reinitialized existing Git repository in D:/gitManual/.git/
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git status
On branch master
Changes not staged for commit:
(use "git add/rm <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
deleted: Readme.md
modified: Text.md
no changes added to commit (use "git add" and/or "git commit -a")


LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git add Text.md
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git commit -m "Text.md file added with Experiment - 1."
[master 11f8cec] Text.md file added with Experiment - 1.
1 file changed, 1 insertion(+)

Experiment 2: Creating and Managing Branches:

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

Solution:

To create a new branch named "feature-branch," switch to the "master" branch, and merge the "feature-branch" into "master" in Git, follow these steps:

1. Make sure you are in the "master" branch by switching to it:
$ git checkout master

2. Create a new branch named "feature-branch" and switch to it:
$ git checkout -b feature-branch
This command will create a new branch called "feature-branch" and switch to it.

3. Make your changes in the "feature-branch" by adding, modifying, or deleting files as needed.

4. Stage and commit your changes in the "feature-branch":
$ git add .
$ git commit -m "Changed from master branch to feature-branch."
Replace "Your commit message for feature-branch" with a descriptive commit message for the changes you made in the "feature-branch."

5. Switch back to the "master" branch:
$ git checkout master

6. Merge the "feature-branch" into the "master" branch:
$ git merge feature-branch
This command will incorporate the changes from the "feature-branch" into the "master" branch.
Now, your changes from the "feature-branch" have been merged into the "master" branch. Your project's history will reflect the changes made in both branches.

```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual ((2bf03ed...))
$ git checkout master
D       Readme.md
Previous HEAD position was 2bf03ed Initial commit : Add Readme.md file
Switched to branch 'master'

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (feature-branch)
$ git add .

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (feature-branch)
$ git commit -m "Changed from master branch to feature-branch."
[feature-branch 6f1d10d] Changed from master branch to feature-branch.
 1 file changed, 2 deletions(-)
 delete mode 100644 Readme.md

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

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git merge feature-branch
Updating 11f8cec..6f1d10d
Fast-forward
 Readme.md | 2 --
 1 file changed, 2 deletions(-)
 delete mode 100644 Readme.md
```

Experiment 3: Creating and Managing Branches:

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

Solution:

To stash your changes, switch branches, and then apply the stashed changes in Git, you can use the following commands:

1. Stash your changes:
$ git stash save "Stash Message"
This command will save your changes in a stash, which acts like a temporary storage for changes that are not ready to be committed.

2. Switch to the desired branch:
$ git checkout target-branch
Replace "target-branch" with the name of the branch you want to switch to.

3. Apply the stashed changes:
$ git stash apply
This command will apply the most recent stash to your current working branch. If you have multiple stashes, you can specify a stash by name or reference (e.g., git stash apply stash@{2}) if needed.
If you want to remove the stash after applying it, you can use git stash pop instead of git stash apply. Remember to replace "Your stash message" and "target-branch" with the actual message you want for your stash and the name of the branch you want to switch to.

```
MINGW64:/d/gitManual

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ nano Text.md

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Text.md

no changes added to commit (use "git add" and/or "git commit -a")

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git stash
Saved working directory and index state WIP on master: 6f1d10d Changed from master branch to feature-br
anch.

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git stash list
stash@{0}: WIP on master: 6f1d10d Changed from master branch to feature-branch.
stash@{1}: WIP on master: 6f1d10d Changed from master branch to feature-branch.

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git stash branch master stash@{0}
fatal: a branch named 'master' already exists

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git stash branch target-branch stash@{0}
Switched to a new branch 'target-branch'
On branch target-branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Text.md

no changes added to commit (use "git add" and/or "git commit -a")
Dropped stash@{0} (0a38d2b4dbdd721900aebfa209fd9d4b6a8071d4)

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (target-branch)
$ git stash apply
error: Your local changes to the following files would be overwritten by merge:
        Text.md
Please commit your changes or stash them before you merge.
Aborting
On branch target-branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Text.md

no changes added to commit (use "git add" and/or "git commit -a")

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (target-branch)
$ git add .
```

```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (target-branch)
$ git commit -m "Stashed in target-branch"
[target-branch 440757e] Stashed in target-branch
 1 file changed, 1 insertion(+), 1 deletion(-)

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (target-branch)
$ git push origin target-branch
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (8/8), 802 bytes | 802.00 KiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'target-branch' on GitHub by visiting:
remote:      https://github.com/subhangidutta23-ship-it/gitManual/pull/new/target-branch
remote:
To https://github.com/subhangidutta23-ship-it/gitManual.git
 * [new branch]      target-branch -> target-branch
```

Experiment 4: Collaboration and Remote Repositories:

Clone a remote Git repository to your local machine.

Solution:

To clone a remote Git repository to your local machine, follow these steps:

1. Open your terminal or command prompt.

2. Navigate to the directory where you want to clone the remote Git repository. You can use the cd command to change your working directory.

3. Use the git clone command to clone the remote repository. Replace <repository_url> with the URL of the remote Git repository you want to clone. For example, if you were cloning a repository from GitHub, the URL might look like this:
$ git clone <repository_url>
Here's a full example:
$ git clone https://github.com/username/repo-name.git
Replace https://github.com/username/repo-name.git with the actual URL of the repository you want to clone.

4. Git will clone the repository to your local machine. Once the process is complete, you will have a local copy of the remote repository in your chosen directory.
You can now work with the cloned repository on your local machine, make changes, and push those changes back to the remote repository as needed.

```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (target-branch)
$ git clone https://github.com/subhangidutta23-ship-it/4GW24CI053.git
Cloning into '4GW24CI053'...
remote: Enumerating objects: 63, done.
remote: Counting objects: 100% (63/63), done.
remote: Compressing objects: 100% (42/42), done.
remote: Total 63 (delta 14), reused 23 (delta 6), pack-reused 0 (from 0)
Receiving objects: 100% (63/63), 415.42 KiB | 1.75 MiB/s, done.
Resolving deltas: 100% (14/14), done.
```

Experiment 5: Collaboration and Remote Repositories:

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

Solution:

To fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch in Git, follow these steps:

1. Open your terminal or command prompt.

2. Make sure you are in the local branch that you want to rebase. You can switch to the branch using the following command, replacing <branch-name> with your actual branch name:
$ git checkout <branch-name>

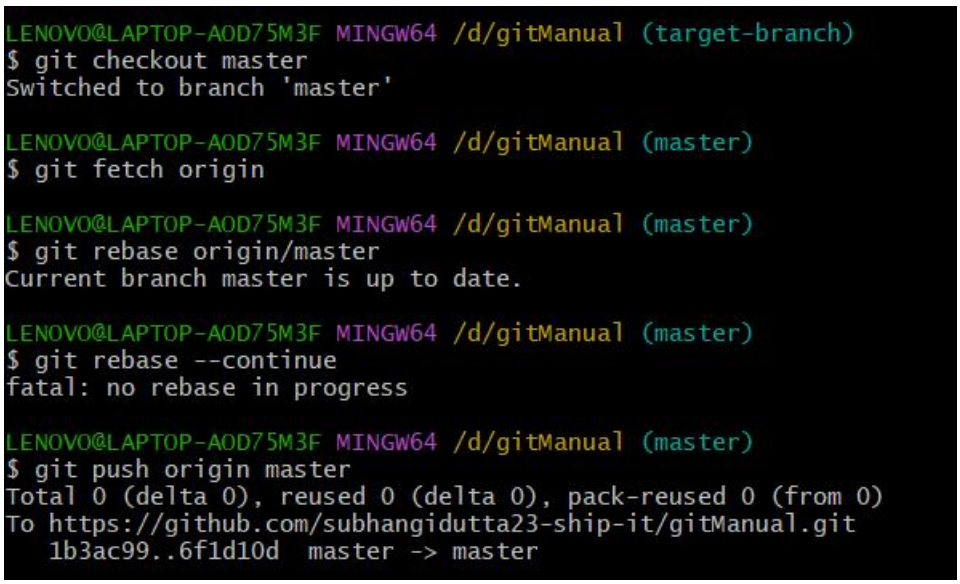
3. Fetch the latest changes from the remote repository. This will update your local repository with the changes from the remote without merging them into your local branch:
$ git fetch origin
Here, origin is the default name for the remote repository. If you have multiple remotes, replace origin with the name of the specific remote you want to fetch from.

4. Once you have fetched the latest changes, rebase your local branch onto the updated remote branch:
$ git rebase origin/<branch-name>
Replace <branch-name> with the name of the remote branch you want to rebase onto. This command will reapply your local commits on top of the latest changes from the remote branch, effectively incorporating the remote changes into your branch history.

5. Resolve any conflicts that may arise during the rebase process. Git will stop and notify you if there are conflicts that need to be resolved. Use a text editor to edit the conflicting files, save the changes, and then continue the rebase with:
$ git rebase –continue

6. After resolving any conflicts and completing the rebase, you have successfully updated your local branch with the latest changes from the remote branch.

7. If you want to push your rebased changes to the remote repository, use the git push command. However, be cautious when pushing to a shared remote branch, as it can potentially overwrite other developers' changes:
$ git push origin <branch-name>
Replace <branch-name> with the name of your local branch. By following these steps, you can keep your local branch up to date with the latest changes from the remote repository and maintain a clean and linear history through rebasing.


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 merge.

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 "feature-branch merged."
```

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."

```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git checkout master
Already on 'master'

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git merge feature-branch -m "feature-branch merged."
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.

Solution:

To create a lightweight Git tag named "v1.0" for a commit in your local repository, you can use the following command:

$ git tag v1.0

This command will create a lightweight tag called "v1.0" for the most recent commit in your current branch. If you want to tag a specific commit other than the most recent one, you can specify the commit's SHA-1 hash after the tag name. For example:

$ git tag v1.0 <commit-SHA>

Replace <commit-SHA> with the actual SHA-1 hash of the commit you want to tag.

```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git tag v1.0
```

```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git tag v2.0

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git tag
v1.0
v2.0
```

```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git log --oneline
6f1d10d (HEAD -> master, tag: v2.0, tag: v1.0, origin/master, origin/HEAD, feature-branch) Changed from
 master branch to feature-branch.
11f8cec Text.md file added with Experiment - 1.
1b3ac99 Text.md added with the contents of Experiment 1.
2bf03ed Initial commit : Add Readme.md file
```

```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git tag v3.0 2bf03ed

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git tag
v1.0
v2.0
v3.0
```

Experiment 8: Advanced Git Operations:

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

Solution:

To cherry-pick a range of commits from "source-branch" to the current branch, you can use the following command:

$ git cherry-pick <start-commit>

Replace <start-commit> with the commit at the beginning of the range.

$ git cherry-pick 2bf03ed

Make sure you are on the branch where you want to apply these changes before running the cherry-pick command.

```
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: Analyzing 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.

Solution:

To view the details of a specific commit, including the author, date, and commit message, you can use the git show or git log command with the commit ID. Here are both options:

1. Using git show:
bash
git show <commit-ID>
Replace <commit-ID> with the actual commit ID you want to view. This command will display detailed information about the specified commit, including the commit message, author, date, and the changes introduced by that commit.
For example:
$ git show 2bf03ed
2. Using git log:
$ git log -n 1 <commit-ID>
The -n 1 option tells Git to show only one commit. Replace <commit-ID> with the actual commit ID. This command will display a condensed view of the specified commit, including its commit message, author, date, and commit ID.
For example:
$ git log -n 1 2bf03ed
Both of these commands will provide you with the necessary information about the specific commit you're interested in.

```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git show 2bf03ed
commit 2bf03eda50026616c4183f27efed40ceac5210bb (tag: v3.0)
Author: subhangidutta <subhnagidutta23@gmail.com>
Date:   Thu Dec 18 11:41:44 2025 +0530

    Initial commit : Add Readme.md file

diff --git a/Readme.md b/Readme.md
new file mode 100644
index 0000000..be1e104
--- /dev/null
+++ b/Readme.md
@@ -0,0 +1,2 @@
+Git Project
+

LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git log -n 1 2bf03ed
commit 2bf03eda50026616c4183f27efed40ceac5210bb (tag: v3.0)
Author: subhangidutta <subhnagidutta23@gmail.com>
Date:   Thu Dec 18 11:41:44 2025 +0530

    Initial commit : Add Readme.md file
```

Experiment 10: Analyzing and Changing Git History:

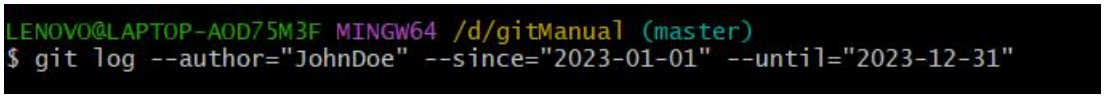
Write the command to list all commits made by the author "JohnDoe" between "2023-01-01" and "2023-12-31."

Solution:

To list all commits made by the author "JohnDoe" between "2023-01-01" and "2023-12-31" in Git, you can use the git log command with the --author and --since and --until options. Here's the command:

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

This command will display a list of commits made by the author "JohnDoe" that fall within the specified date range, from January 1, 2023, to December 31, 2023. Make sure to adjust the author name and date range as needed for your specific use case.



No output occurs when the author name or date range does not match any commits in the repository.

Experiment 11: Analyzing and Changing Git History:

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

Solution:

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.

```
LENOVO@LAPTOP-AOD75M3F MINGW64 /d/gitManual (master)
$ git log -n 5
commit 15e2efaf9a703f65ecea786354c4110032089a6f (HEAD -> master)
Author: subhangidutta <subhnagidutta23@gmail.com>
Date:   Thu Dec 18 11:41:44 2025 +0530

    Initial commit : Add Readme.md file

commit 6f1d10d6588ee1f410121f441fc205f14c541704 (tag: v2.0, tag: v1.0, origin/master, origin/HEAD)
Author: subhangidutta <subhnagidutta23@gmail.com>
Date:   Tue Jan 6 03:51:39 2026 +0530

    Changed from master branch to feature-branch.

commit 11f8cec9d88595142991136031ade04db002f6ff
Author: subhangidutta <subhnagidutta23@gmail.com>
Date:   Tue Jan 6 03:22:06 2026 +0530

    Text.md file added with Experiment - 1.

commit 1b3ac998c8492b9f356d8d7f6d7362f13c8cc733
Author: subhangidutta <subhnagidutta23@gmail.com>
Date:   Tue Jan 6 02:53:48 2026 +0530

    Text.md added with the contents of Experiment 1.

commit 2bf03eda50026616c4183f27efed40ceac5210bb (tag: v3.0)
Author: subhangidutta <subhnagidutta23@gmail.com>
Date:   Thu Dec 18 11:41:44 2025 +0530

    Initial commit : Add Readme.md file
```

Experiment 12: Analyzing and Changing Git History:

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

Solution:

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.

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

```
E325: ATTENTION
Found a swap file by the name "/d/gitManual/.git/.COMMIT_EDITMSG.swp"
         owned by: LENOVO   dated: Tue Jan 06 02:58:52 2026
        file name: /d/gitManual/.git/COMMIT_EDITMSG
         modified: YES
        user name: LENOVO   host name: LAPTOP-AOD75M3F
       process ID: 804
While opening file "/d/gitManual/.git/COMMIT_EDITMSG"
             dated: Tue Jan 06 05:26:21 2026
      NEWER than swap file!

(1) Another program may be editing the same file.  If this is the case,
    be careful not to end up with two different instances of the same
    file when making changes.  Quit, or continue with caution.
(2) An edit session for this file crashed.
    If this is the case, use ":recover" or "vim -r /d/gitManual/.git/COMMIT_EDITMSG"
    to recover the changes (see ":help recovery").
    If you did this already, delete the swap file "/d/gitManual/.git/.COMMIT_EDITMSG.swp"
    to avoid this message.

Swap file "/d/gitManual/.git/.COMMIT_EDITMSG.swp" already exists!
[O]pen Read-Only, (E)dit anyway, (R)ecover, (D)elete it, (Q)uit, (A)bort:
Swap file "/d/gitManual/.git/.COMMIT_EDITMSG.swp" already exists!
[O]pen Read-Only, (E)dit anyway, (R)ecover, (D)elete it, (Q)uit, (A)bort:
```

```
[master ae80ba5] Revert "Initial commit : Add Readme.md file"
 1 file changed, 2 deletions(-)
 delete mode 100644 Readme.md
```

D. Appendix:

Repository Link: https://github.com/subhangidutta23-ship-it/4GW24CI053.git