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

MOHAMMED AYESHA TAHREEM

4GW24CI027

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

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:

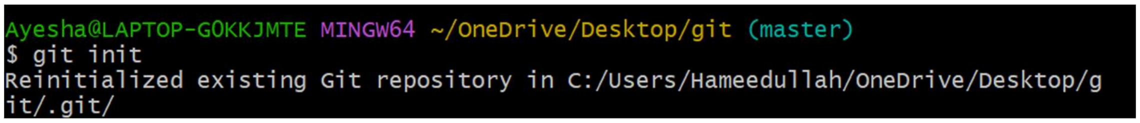
To initialize a new Git repository in a directory, create a new file, add it to the staging area,

and commit the changes with an appropriate commit message, follow these steps:

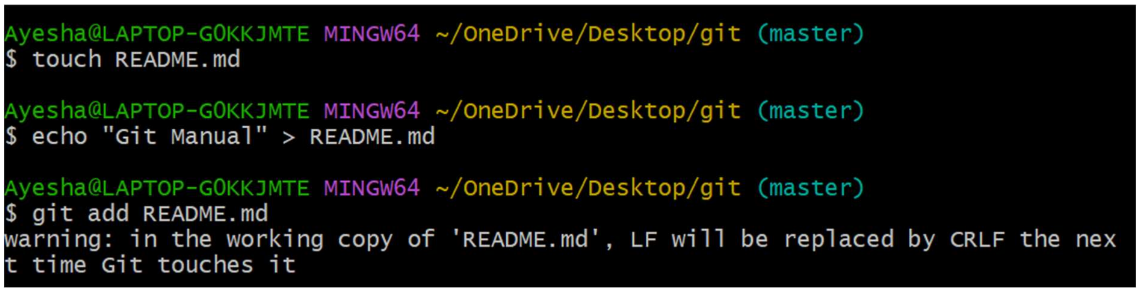
1. Open your terminal and navigate to the directory where you want to create the Git

repository.

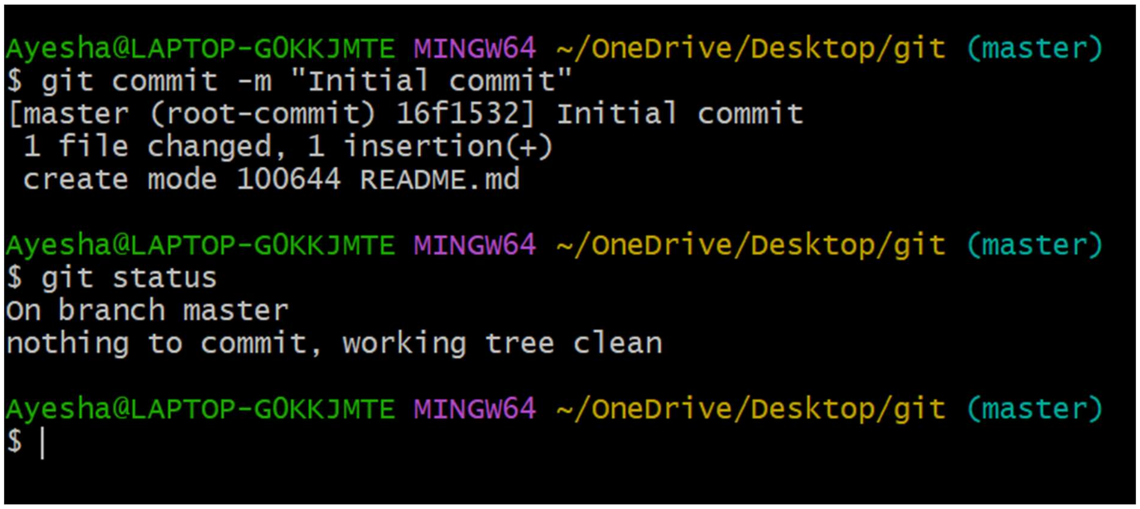
2. Initialize a new Git repository in that directory



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ git init
Reinitialized existing Git repository in C:/Users/Hameedullah/OneDrive/Desktop/g
it/.git/
```



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ touch README.md

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ echo "Git Manual" > README.md

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the nex
t time Git touches it
```



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ git commit -m "Initial commit"
[master (root-commit) 16f1532] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ git status
On branch master
nothing to commit, working tree clean

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ |
```

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. Configure user identity (username and email ID):

```
$ git config user.email "your_email@gmail.com"
$ git config user.name "your_username"
```

4.Create a new file in the directory. For example, let's create a file named README.md.
You can use any text editor or command-line tools to create the file:

```
$ touch README.md
```

5.Add content to the file:

```
$ echo "Git Manual" > README.md
```

6.Check the status of the repository:

```
$ git status
```

7.Add the newly created file to the staging area:

```
$ git add README.md
```

This command stages the file for the upcoming commit.

8.Commit the changes with an appropriate commit message:

```
$ git commit -m "Initial commit"
```

Your commit message should briefly describe the purpose or nature of the changes you made.

9.**Verify the commit history:**

```
$ git log --oneline
```

10.View the contents of the file

```
$ cat README.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.

Experiment 2.

Creating and Managing Branches:

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

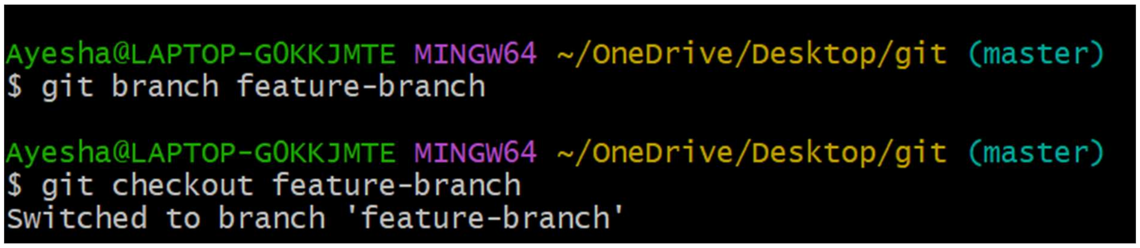
Merge the "feature-branch" into "master."

Solution:

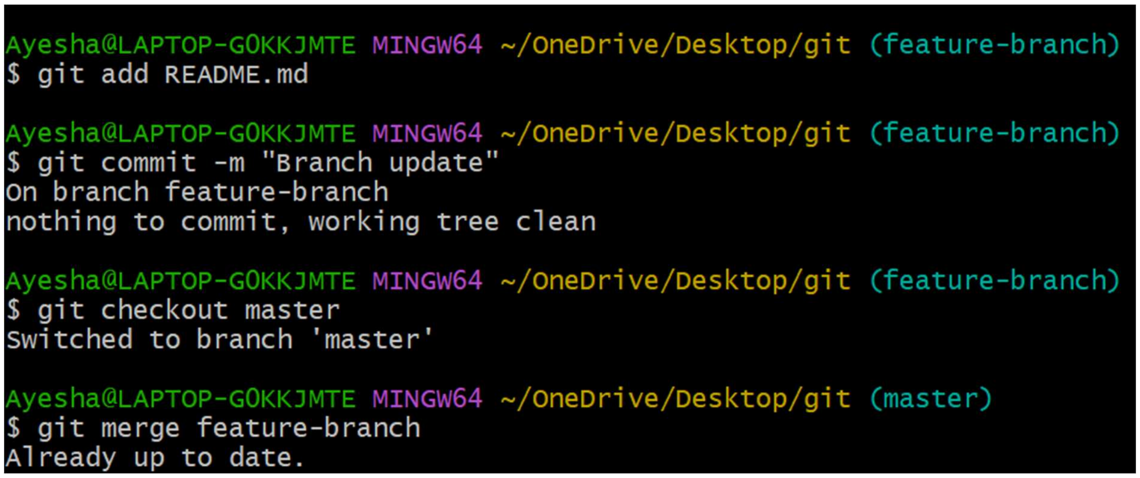
To create a new branch named "feature-branch," switch to the "master" branch, and merge the

"feature-branch" into "master" in Git, follow these steps:

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



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ git branch feature-branch

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ git checkout feature-branch
Switched to branch 'feature-branch'
```



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (feature-branch)
$ git add README.md

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (feature-branch)
$ git commit -m "Branch update"
On branch feature-branch
nothing to commit, working tree clean

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (feature-branch)
$ git checkout master
Switched to branch 'master'

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ git merge feature-branch
Already up to date.
```

1. **Create a new branch named feature-branch:**

```
$ git branch feature-branch
```

2. **Switch to the newly created branch:**

```
$ git checkout feature-branch
```

3. **Make changes in the branch and add the file to the staging area:**

```
$ git add README.md
```

4. **Commit the changes in the feature branch:**

```
$ git commit -m "Branch update"
```

5. **Switch back to the master branch:**

```
$ git checkout master
```

6. **Merge the feature branch into the master branch:**

```
$ git merge feature-branch
```

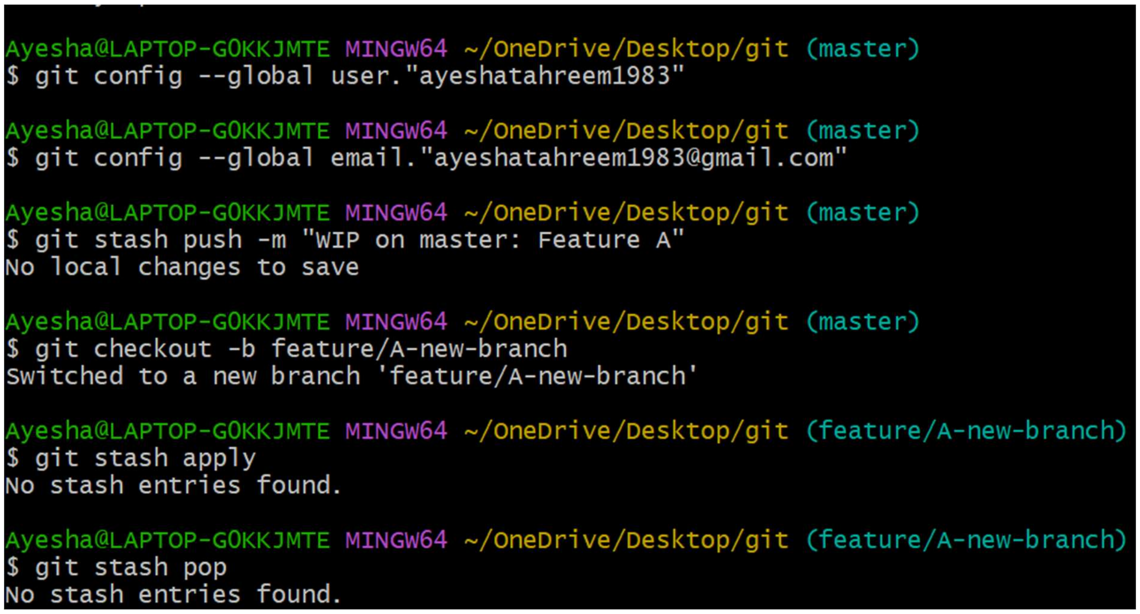
After these steps, the changes from the feature-branch are successfully merged into the master branch.

Program 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:



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ git config --global user."ayeshatahreem1983"

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ git config --global email."ayeshatahreem1983@gmail.com"

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ git stash push -m "WIP on master: Feature A"
No local changes to save

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (master)
$ git checkout -b feature/A-new-branch
Switched to a new branch 'feature/A-new-branch'

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (feature/A-new-branch)
$ git stash apply
No stash entries found.

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (feature/A-new-branch)
$ git stash pop
No stash entries found.
```

1.Configure global username and email ID:

```
$ git config --global user.name "your_username"
$ git config --global user.email your_email@gmail.com
```

2. Stash the current changes with a message:

```
$ git stash push -m "WIP on master: Feature A"
```

This command saves the uncommitted changes temporarily without committing them.

3. Create and switch to a new branch:

```
$ git checkout -b feature/A-new-branch
```

4. Apply the stashed changes to the current branch:

```
$ git stash apply
```

5. Alternatively, apply and remove the stash entry:

```
$ git stash pop
```

After executing these commands, the stashed changes are successfully applied to the new branch.

Experiment 4.

Collaboration and Remote Repositories:

Clone a remote Git repository to your local machine.

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

1. Open your terminal or command prompt.

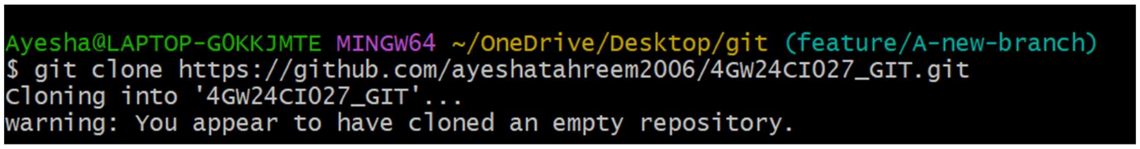
2. Navigate to the directory where you want to clone the remote Git repository. You can

use the cd command to change your working directory.

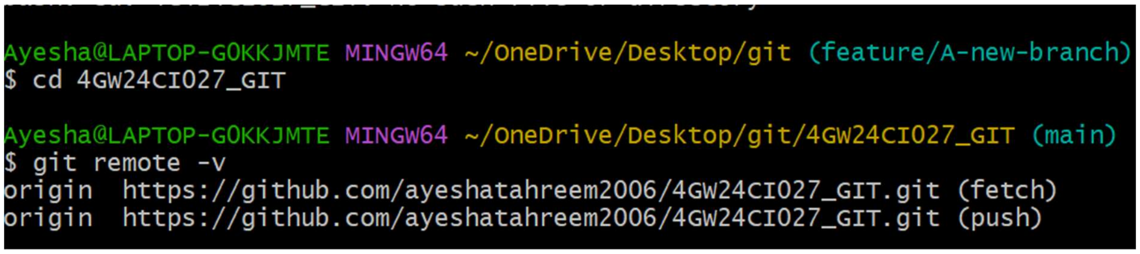
3. Use the git clone command to clone the remote repository. Replace <repository_url>

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:



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (feature/A-new-branch)
$ git clone https://github.com/ayeshatahreem2006/4GW24CI027_GIT.git
Cloning into '4GW24CI027_GIT'...
warning: You appear to have cloned an empty repository.
```



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git (feature/A-new-branch)
$ cd 4GW24CI027_GIT

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git/4GW24CI027_GIT (main)
$ git remote -v
origin  https://github.com/ayeshatahreem2006/4GW24CI027_GIT.git (fetch)
origin  https://github.com/ayeshatahreem2006/4GW24CI027_GIT.git (push)
```

1. **Clone the repository from GitHub using the repository URL:**

   $ git clone https://github.com/your_username/your_repository.git

2. **Navigate into the cloned repository directory:**

   $ cd your_repository

3. Verify the remote repository URL:

```
$ git remote -v
```

After executing these commands, the GitHub repository is successfully cloned to the local system, and the remote URL is verified.

Experiment 5.

Collaboration and Remote Repositories:

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

onto the updated remote branch.

To fetch the latest changes from a remote repository and rebase your local branch onto the

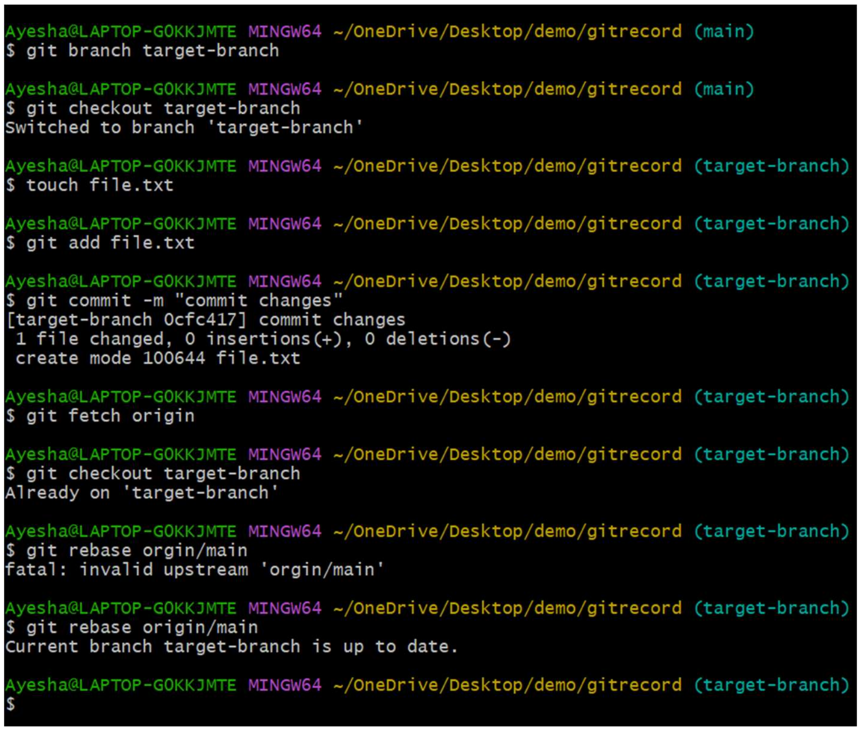
updated remote branch in Git, follow these steps:

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:



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (main)
$ git branch target-branch

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (main)
$ git checkout target-branch
Switched to branch 'target-branch'

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (target-branch)
$ touch file.txt

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (target-branch)
$ git add file.txt

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (target-branch)
$ git commit -m "commit changes"
[target-branch 0cfc417] commit changes
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file.txt

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (target-branch)
$ git fetch origin

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (target-branch)
$ git checkout target-branch
Already on 'target-branch'

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (target-branch)
$ git rebase orgin/main
fatal: invalid upstream 'orgin/main'

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (target-branch)
$ git rebase origin/main
Current branch target-branch is up to date.

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (target-branch)
$
```

1. Create a new branch named target-branch:

```
$ git branch target-branch
```

2. Switch to the newly created branch:

```
$ git checkout target-branch
```

3. Create a new file in the branch:

```
$ touch file.txt
```

4. Add the file to the staging area:

```
$ git add file.txt
```

5. Commit the changes:

```
$ git commit -m "Commit changes"
```

6. Fetch the latest changes from the remote repository:

```
$ git fetch origin
```

7. Ensure you are on the target branch:

```
$ git checkout target-branch
```

8. Rebase the target branch with the main branch:

```
$ git rebase origin/main
```

This command reapplies the commits of the current branch on top of the latest commits from the main branch.

After executing these steps, the target-branch is successfully rebased with the main branch and is up to date.

Experiment 6.

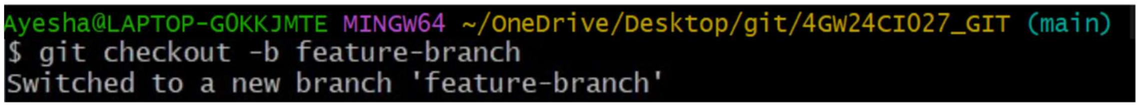
Collaboration and Remote Repositories:

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

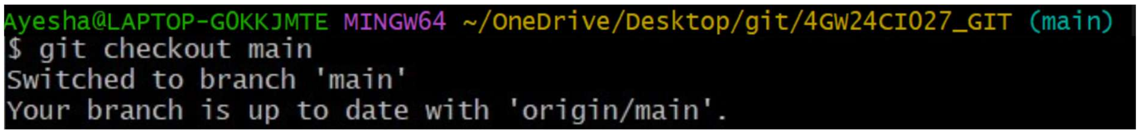
Solution:

To merge the "feature-branch" into "master" in Git while providing a custom commit message

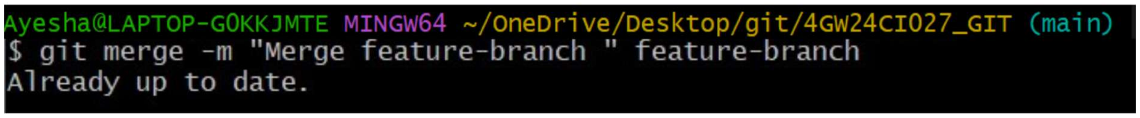
for the merge, you can use the following command:



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git/4GW24CI027_GIT (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
```



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git/4GW24CI027_GIT (main)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/git/4GW24CI027_GIT (main)
$ git merge -m "Merge feature-branch " feature-branch
Already up to date.
```

1: Create and Switch to a New Branch

```
git checkout -b feature-branch
```

2: Make a Change and Commit

```
# Create a dummy file or edit an existing one
echo "Hello World" > record.txt
# Stage and commit the change
git add .
git commit -m "Add new changes to feature branch"
```

3: Switch Back to the Main Branch

Before you can merge the feature into the main branch, you must move back to main.

```
git checkout main
```

4: Merge the Feature Branch

This pulls the changes from feature-branch into your current branch (main).

```
git merge -m "Merge feature-branch" feature-branch
```

Experiment 7.

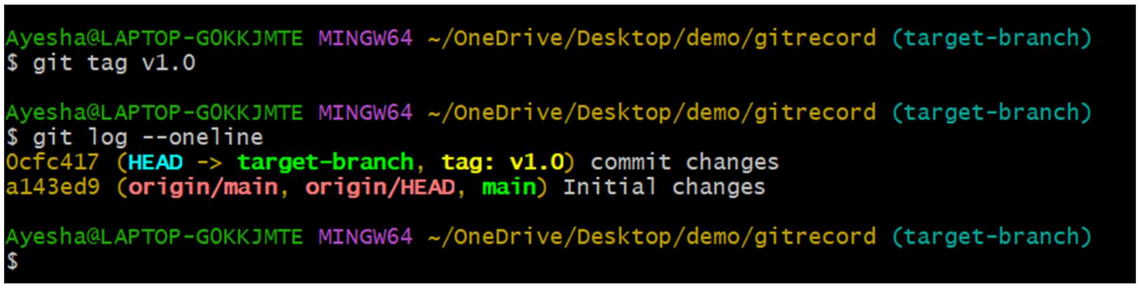
Git Tags and Releases:

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:



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (target-branch)
$ git tag v1.0

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (target-branch)
$ git log --oneline
0cfc417 (HEAD -> target-branch, tag: v1.0) commit changes
a143ed9 (origin/main, origin/HEAD, main) Initial changes

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~/OneDrive/Desktop/demo/gitrecord (target-branch)
$
```

Step 1: Create a Lightweight Tag

A lightweight tag is a simple pointer to a specific commit, often used to mark a version or milestone. The screenshot shows a tag named v1.0 being created on the current commit.

```
git tag v1.0
```

Step 2: View History in a Condensed Format

The git log --oneline command displays the commit history as a compact list. It shows the abbreviated 7-character commit hash and the first line of the commit message for each entry.

```
git log --oneline
```

Experiment 8.

Advanced Git Operations:

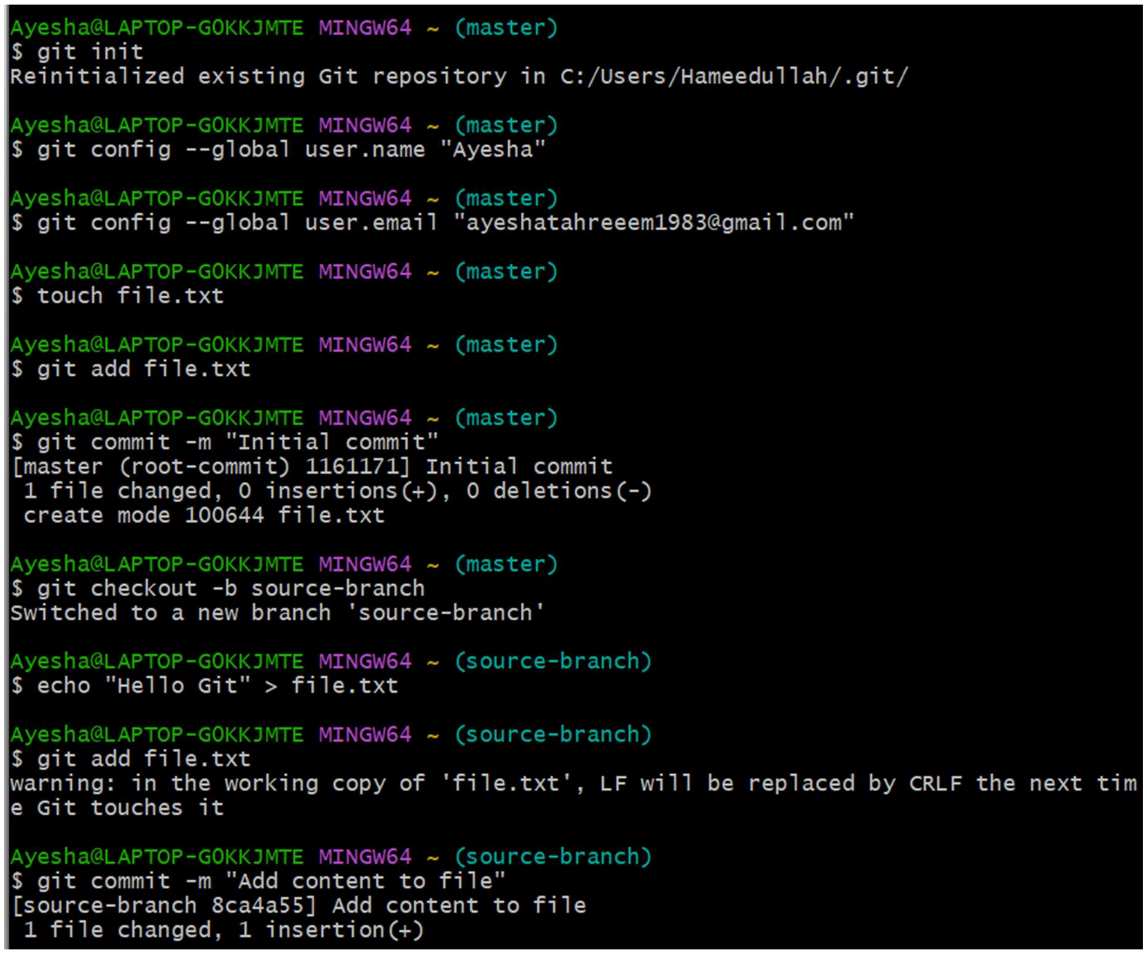
Write the command to cherry-pick a range of commits from "source-branch" to the

current branch.

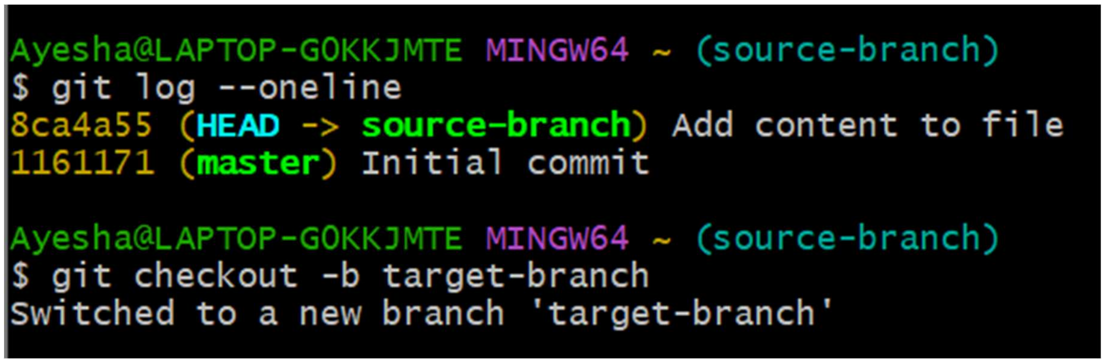
Solution:

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

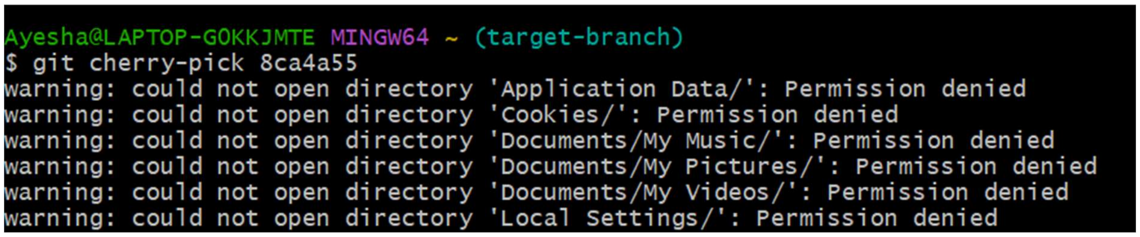
the following command:



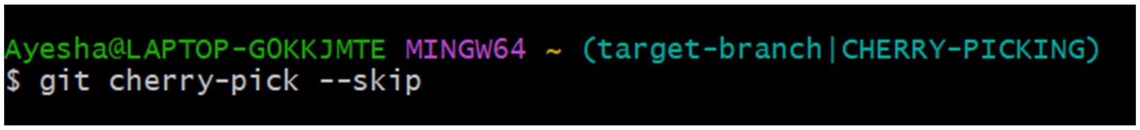
```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (master)
$ git init
Reinitialized existing Git repository in C:/Users/Hameedullah/.git/

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (master)
$ git config --global user.name "Ayesha"

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (master)
$ git config --global user.email "ayeshatahreeem1983@gmail.com"

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (master)
$ touch file.txt

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (master)
$ git add file.txt

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (master)
$ git commit -m "Initial commit"
[master (root-commit) 1161171] Initial commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file.txt

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (master)
$ git checkout -b source-branch
Switched to a new branch 'source-branch'

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (source-branch)
$ echo "Hello Git" > file.txt

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (source-branch)
$ git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next tim
e Git touches it

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (source-branch)
$ git commit -m "Add content to file"
[source-branch 8ca4a55] Add content to file
 1 file changed, 1 insertion(+)
```



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (source-branch)
$ git log --oneline
8ca4a55 (HEAD -> source-branch) Add content to file
1161171 (master) Initial commit

Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (source-branch)
$ git checkout -b target-branch
Switched to a new branch 'target-branch'
```



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (target-branch)
$ git cherry-pick 8ca4a55
warning: could not open directory 'Application Data/': Permission denied
warning: could not open directory 'Cookies/': Permission denied
warning: could not open directory 'Documents/My Music/': Permission denied
warning: could not open directory 'Documents/My Pictures/': Permission denied
warning: could not open directory 'Documents/My Videos/': Permission denied
warning: could not open directory 'Local Settings/': Permission denied
```



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (target-branch|CHERRY-PICKING)
$ git cherry-pick --skip
```

Part 1: Initial Setup and Branching

1. Initialize the repository:

git init

2. Configure user identity:

git config --global user.name "Ayesha"

git config --global user.email ayeshatahreeem1983@gmail.com

3. Create and commit the first file:

touch file.txt

git add file.txt

git commit -m "Initial commit"

4. Create and switch to a new branch

git checkout -b source-branch

5. Modify the file and commit on the new branch

echo "Hello Git" > file.txt

```
git add file.txt
git commit -m "Add content to file"
```

Part 2: Cherry-Picking and Logs

1. View history to find a commit hash:

```
git log –oneline
```

2. Create a target branch for the cherry-pick:

```
git checkout -b target-branch
```

3. Attempt to cherry-pick a specific commit:

```
git cherry-pick 8ca4a55
```

4. Skip the cherry-pick if issues occur:

```
git cherry-pick --skip
```

Experiment 9.

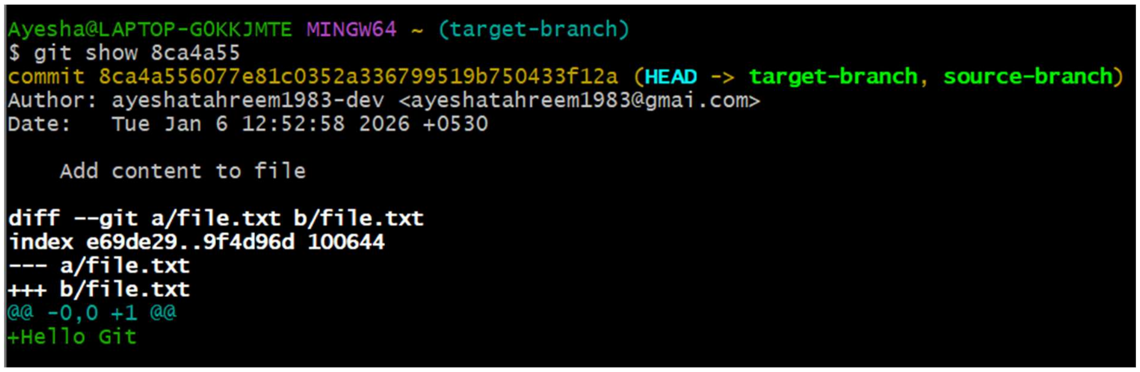
Analysing and Changing Git History:

Given a commit ID, how would you use Git to view the details of that specific

commit, including the author, date, and commit message?

<u>Solution:</u>

To view the details of a specific commit, including the author, date, and commit message



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (target-branch)
$ git show 8ca4a55
commit 8ca4a556077e81c0352a336799519b750433f12a (HEAD -> target-branch, source-branch)
Author: ayeshatahreem1983-dev <ayeshatahreem1983@gmai.com>
Date:   Tue Jan 6 12:52:58 2026 +0530

    Add content to file

diff --git a/file.txt b/file.txt
index e69de29..9f4d96d 100644
--- a/file.txt
+++ b/file.txt
@@ -0,0 +1 @@
+Hello Git
```

1. Using git show:

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.

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."

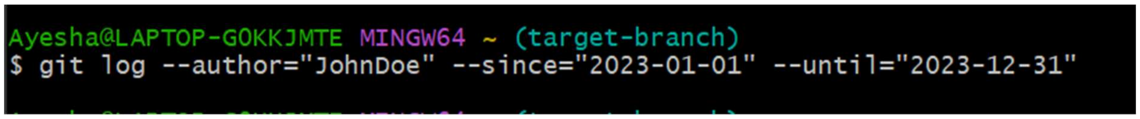
<u>Solution:</u>

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"
```



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (target-branch)
$ 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

<u>No output occurs when the author name or date range does not match any commits in the repository</u>

Experiment 11.

Analysing and Changing Git History

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

Solution:

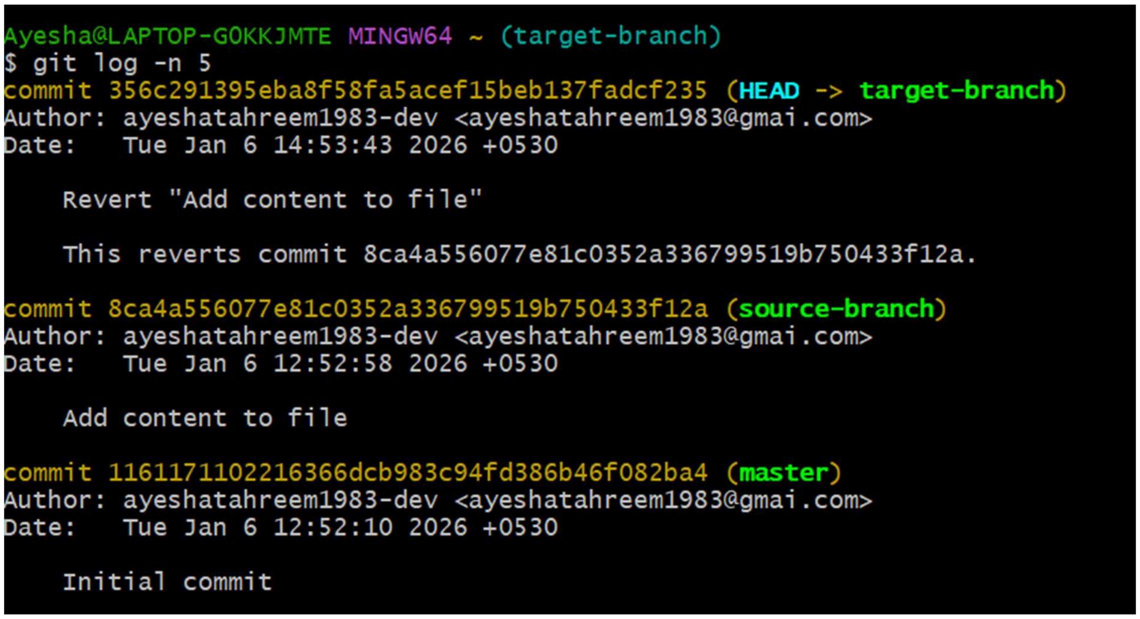
To display the last five commits in a Git repository's history, you can use the git log command

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



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (target-branch)
$ git log -n 5
commit 356c291395eba8f58fa5acef15beb137fadcf235 (HEAD -> target-branch)
Author: ayeshatahreem1983-dev <ayeshatahreem1983@gmai.com>
Date:   Tue Jan 6 14:53:43 2026 +0530

    Revert "Add content to file"

    This reverts commit 8ca4a556077e81c0352a336799519b750433f12a.

commit 8ca4a556077e81c0352a336799519b750433f12a (source-branch)
Author: ayeshatahreem1983-dev <ayeshatahreem1983@gmai.com>
Date:   Tue Jan 6 12:52:58 2026 +0530

    Add content to file

commit 1161171102216366dcb983c94fd386b46f082ba4 (master)
Author: ayeshatahreem1983-dev <ayeshatahreem1983@gmai.com>
Date:   Tue Jan 6 12:52:10 2026 +0530

    Initial commit
```

Experiment 12.

Analysing and Changing Git History

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

Solution:

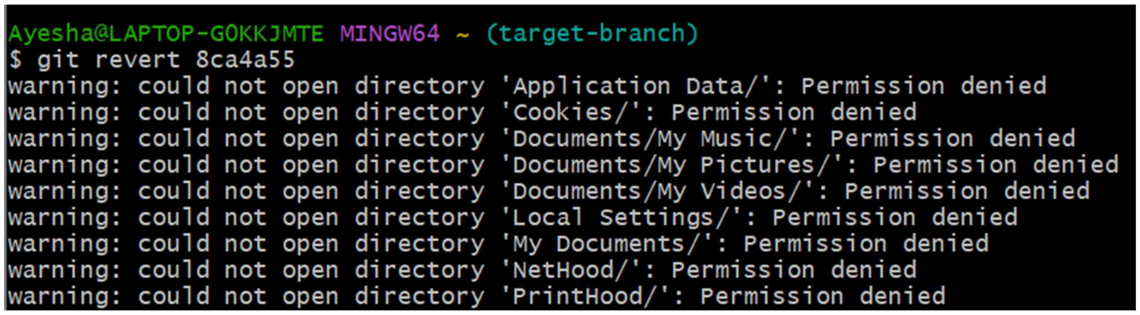
$ git revert abc123

Replace "abc123" with the actual commit ID that you want to revert. After running this

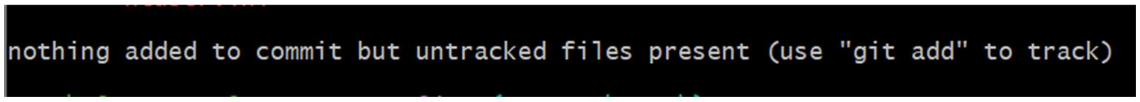
command, Git will create a new commit that negates the changes introduced by the specified

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 change



```
Ayesha@LAPTOP-GOKKJMTE MINGW64 ~ (target-branch)
$ git revert 8ca4a55
warning: could not open directory 'Application Data/': Permission denied
warning: could not open directory 'Cookies/': Permission denied
warning: could not open directory 'Documents/My Music/': Permission denied
warning: could not open directory 'Documents/My Pictures/': Permission denied
warning: could not open directory 'Documents/My Videos/': Permission denied
warning: could not open directory 'Local Settings/': Permission denied
warning: could not open directory 'My Documents/': Permission denied
warning: could not open directory 'NetHood/': Permission denied
warning: could not open directory 'PrintHood/': Permission denied
```



```
nothing added to commit but untracked files present (use "git add" to track)
```