VISVESVARAYA TECHNOLOGICAL UNIVERSITY

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

Lab Experiment Record

Project Management with Git [BCSL58C]

Submitted in partial fulfillment towards AEC of 3rd semester of

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

Submitted by

CHANDRAKALA S
4GW24CI009

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

Project Management with Git		Semester	3
Course Code	BCS358C	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0: 0 : 2: 0	SEE Marks	50
Credits	01	Exam Marks	100
Examination type (SEE)	Practical		

Course objectives:

- .To familiar with basic command of Git
- To create and manage branches
- To understand how to collaborate and work with Remote Repositories
- To familiar with virion controlling commands

Sl.NO	Experiments
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 branch.
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".

Course outcomes (Course Skill Set):

At the end of the course the student will be able to:

- Use the basics commands related to git repository
- Create and manage the branches
- Apply commands related to Collaboration and Remote Repositories
- Use the commands related to Git Tags, Releases and advanced git operations
- Analyse and change the git history

Course Contents :

1. *Git Basics.*
2. *Git Installation*
3. *Git Basic Commands*
4. *Experiments*

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.*

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

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 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

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"

Introduction

Git is a distributed version control system (VCS) that is widely used for tracking changes in source code during software development

It was created by Linus Torvalds in 2005 and has since become the de facto standard for version control in the software development industry.

Git is an essential tool in software development and for many other collaborative and version controlled tasks.

Program 1: *Setting Up and Basic Commands*

Commands Used:

1. Initialize a new Git repository:
   - Git clone
   - cd git_lab
   - git init
2. Create a new file:
   - touch lab1.txt
3. Add the file to the staging area:
   - git add lab1.txt
4. Commit the changes with a message:
   - git commit -m "Initial commit - adding a new file"
   - git push origin main

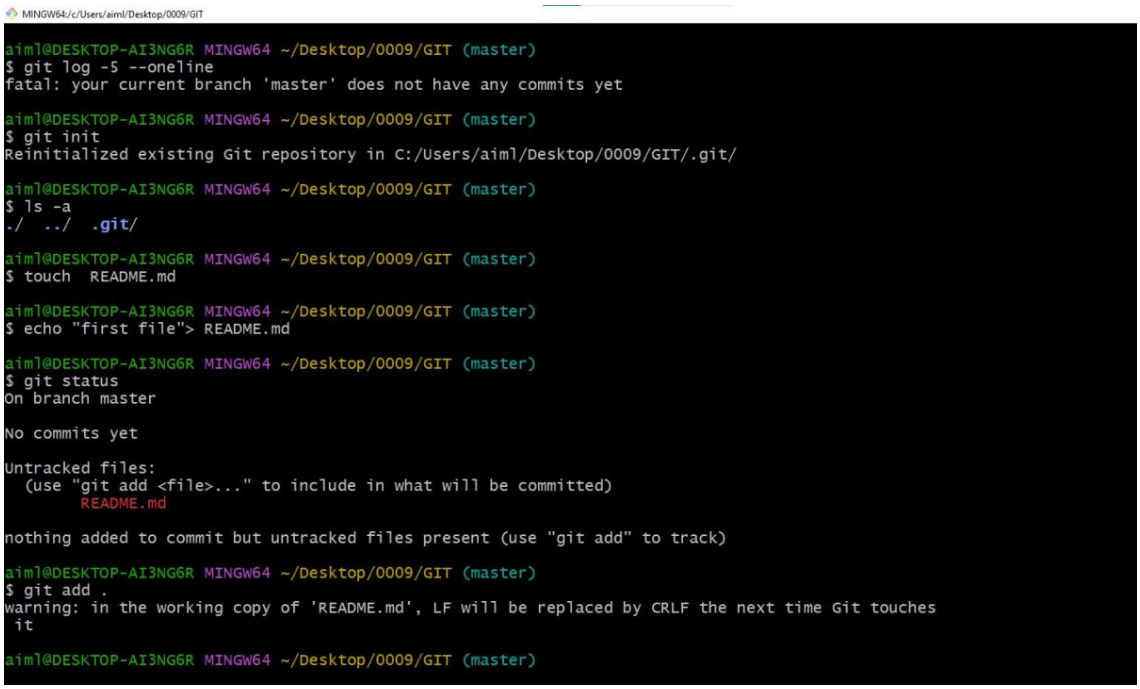
use the ‘status’ command so we can see what git is tracking.
git status

Add the new file using the ‘add’ command.

Git add .

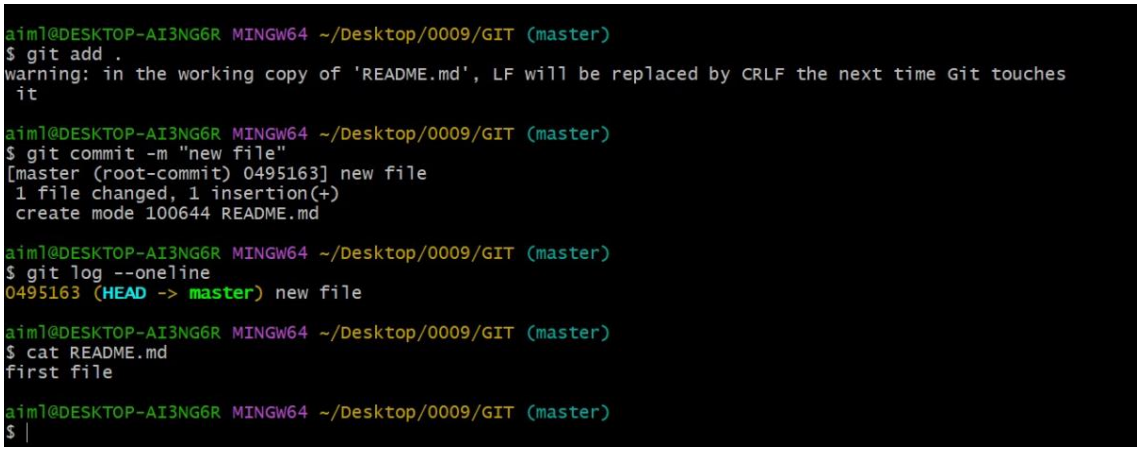
The ‘.’ Adds all the files in your folder in this case only the README.md file

git clone: https://github.com/chandrakalashashidhar25-maker/4GW24CI009.git



Commit the changes using ‘commit’ command.

git commit -m "Initial Commit"



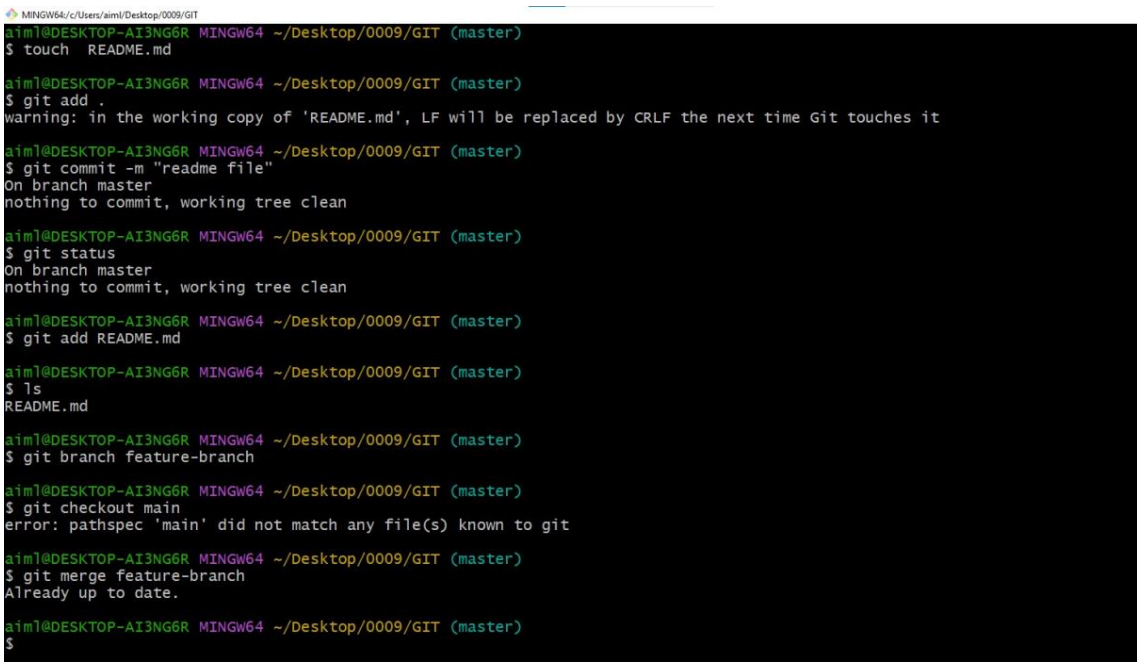
You can confirm the commit using the ‘log –oneline’ message and view the content of the file using ‘cat’ command.

git log --oneline

cat README.md

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

The command is: git branch feature-branch



‘checkout’ command.

The command is: git checkout main

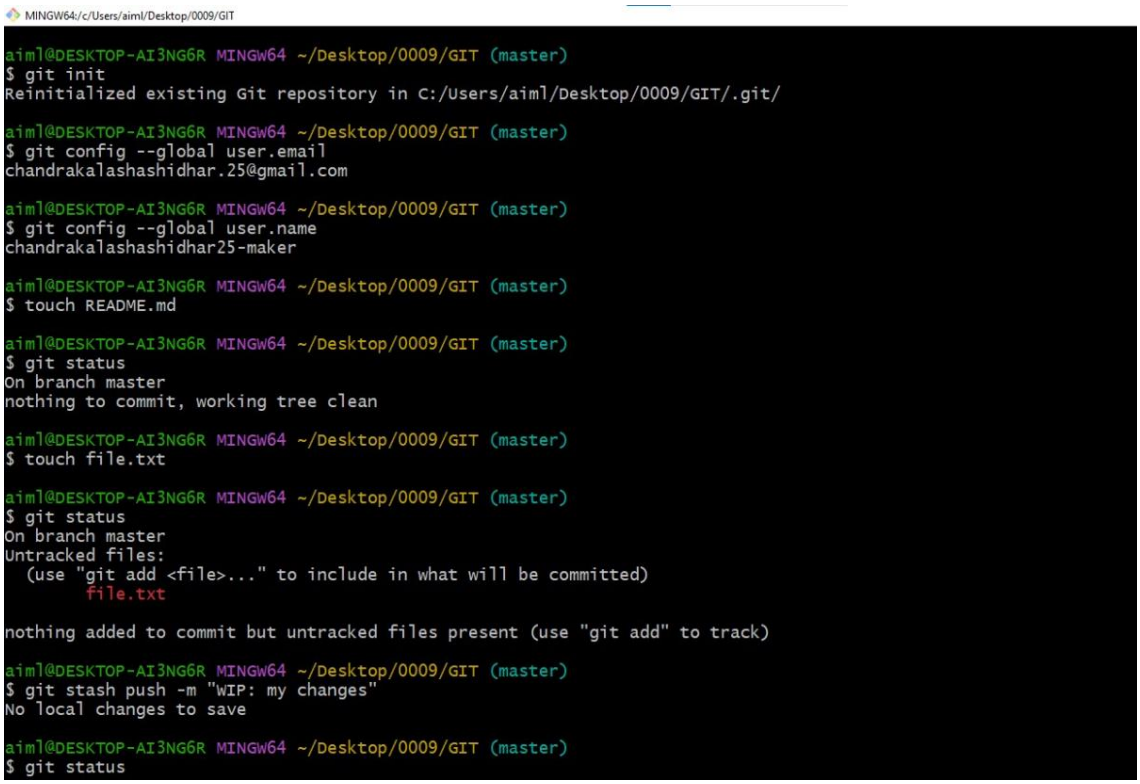
To go back to the feature-branch we have to use: git checkout feature-branch

we use the ‘merge’ command to merge both the branches.

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

git stash push -m "WIP: my changes"

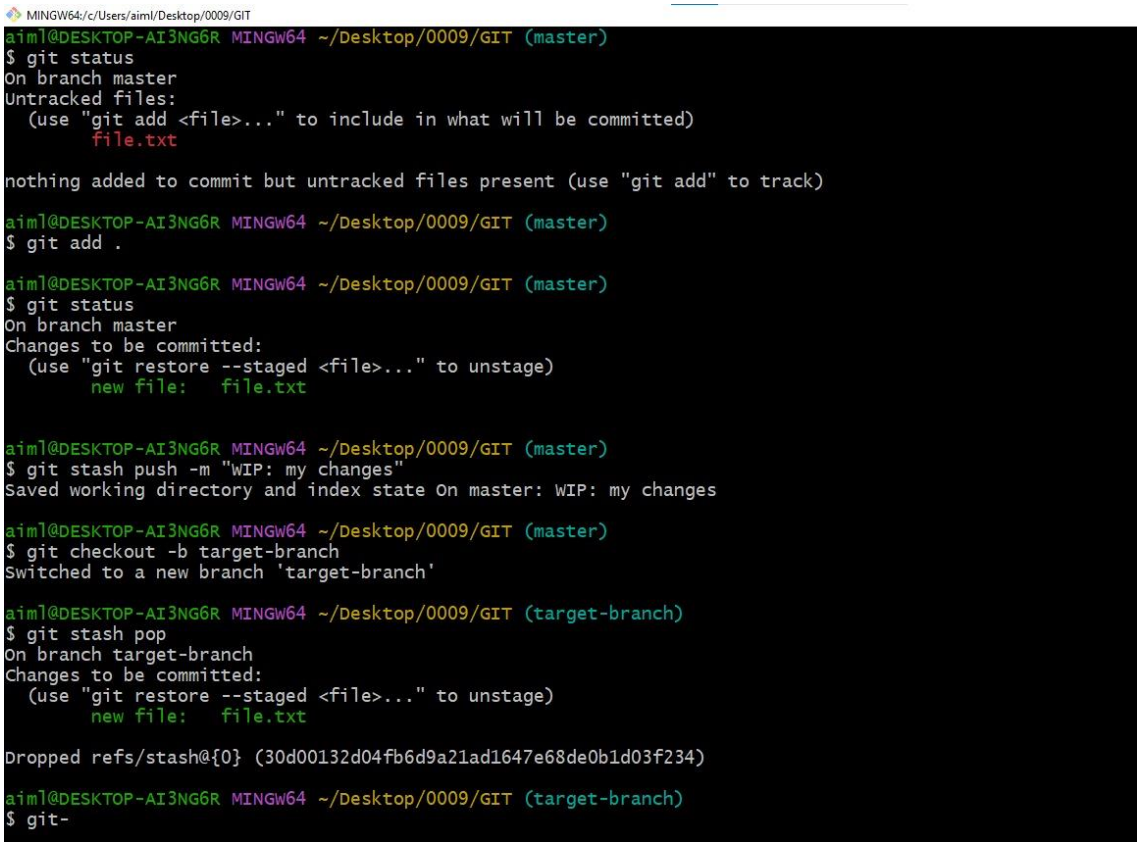
(WIP- work in progress)



git checkout -b target-branch

restore the changes we have made in our previous branch to this branch,

git stash pop

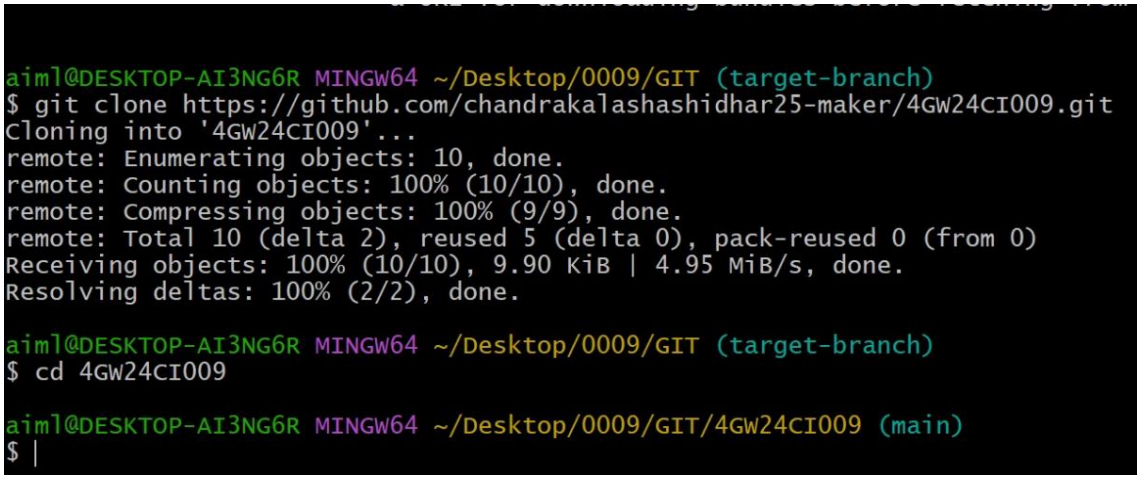

MINGW64:/c/Users/aiml/Desktop/0009/GIT
aiml@DESKTOP-AI3NG6R MINGW64 ~/Desktop/0009/GIT (master)
$ git status
On branch master
Untracked files:
(use "git add <file>..." to include in what will be committed)
file.txt
nothing added to commit but untracked files present (use "git add" to track)
aiml@DESKTOP-AI3NG6R MINGW64 ~/Desktop/0009/GIT (master)
$ git add .
aiml@DESKTOP-AI3NG6R MINGW64 ~/Desktop/0009/GIT (master)
$ git status
On branch master
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
new file: file.txt
aiml@DESKTOP-AI3NG6R MINGW64 ~/Desktop/0009/GIT (master)
$ git stash push -m "WIP: my changes"
Saved working directory and index state On master: WIP: my changes
aiml@DESKTOP-AI3NG6R MINGW64 ~/Desktop/0009/GIT (master)
$ git checkout -b target-branch
Switched to a new branch 'target-branch'
aiml@DESKTOP-AI3NG6R MINGW64 ~/Desktop/0009/GIT (target-branch)
$ git stash pop
On branch target-branch
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
new file: file.txt
Dropped refs/stash@{0} (30d00132d04fb6d9a21ad1647e68de0b1d03f234)
aiml@DESKTOP-AI3NG6R MINGW64 ~/Desktop/0009/GIT (target-branch)
$ git-

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

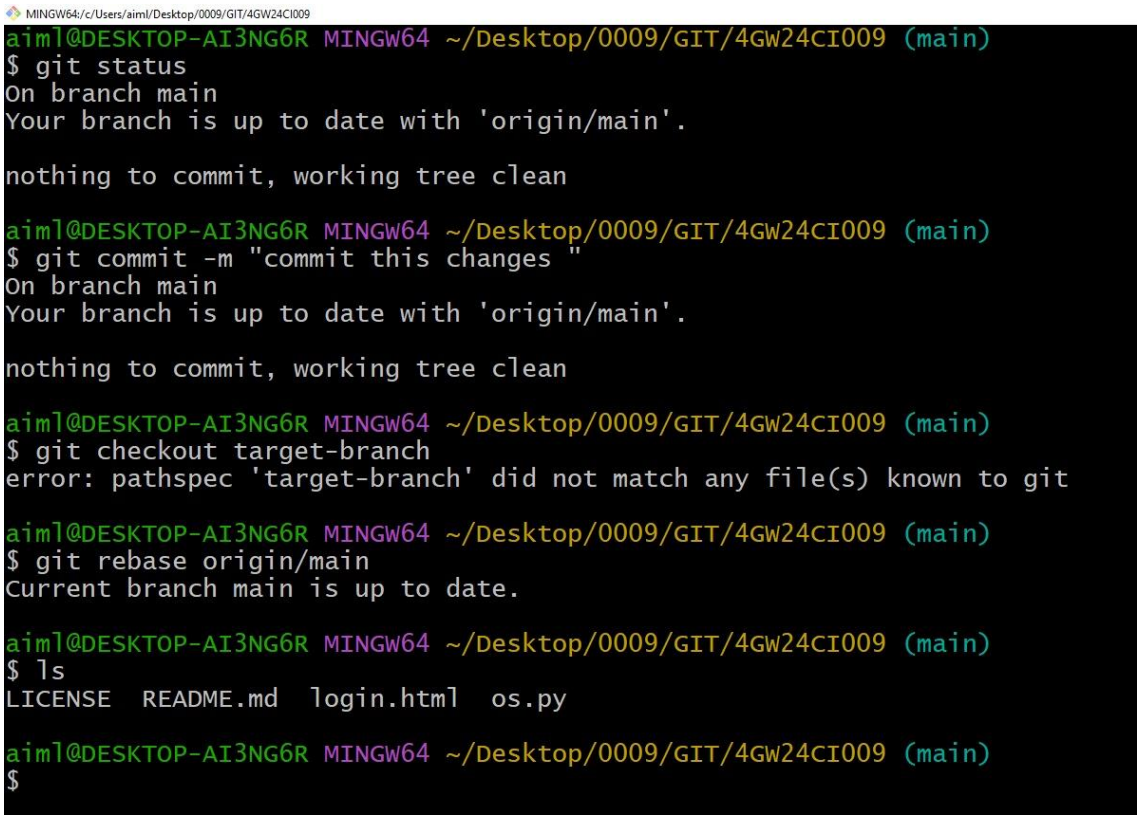
git clone: https://github.com/chandrakalashashidhar25-maker/4GW24CI009.git

can change our directory to the name of our repository.

cd 4GW24CI009



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



git fetch origin

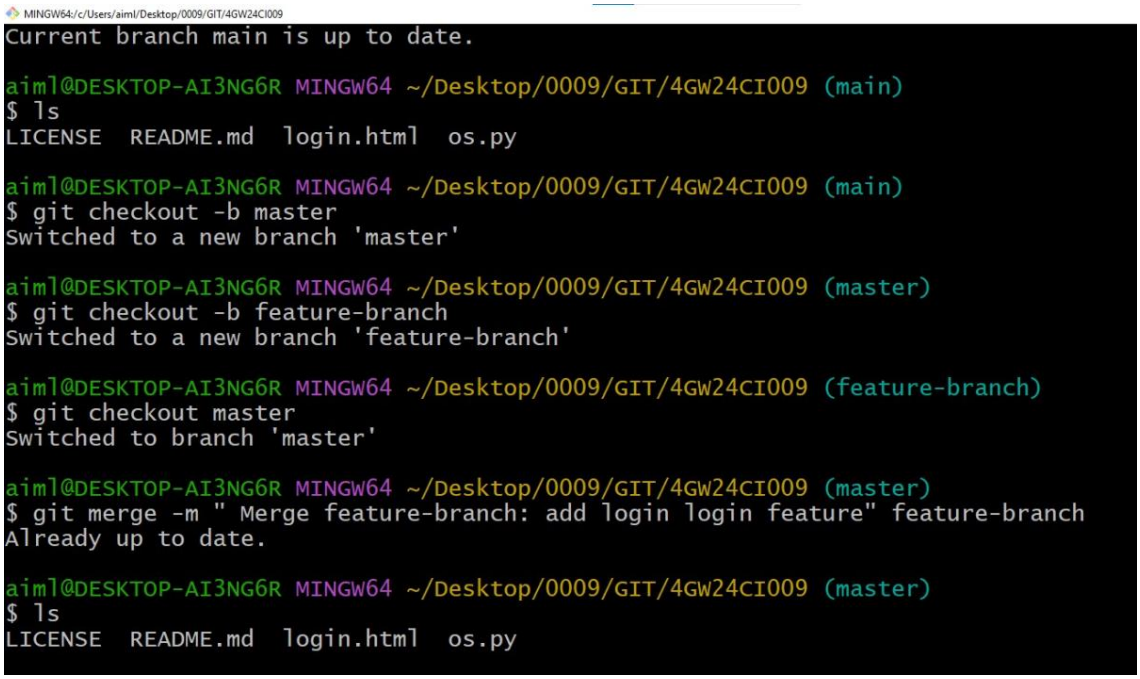
switch to the target-branch. Then rebase local branch onto remote.

git checkout target-branch

git rebase origin/main

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

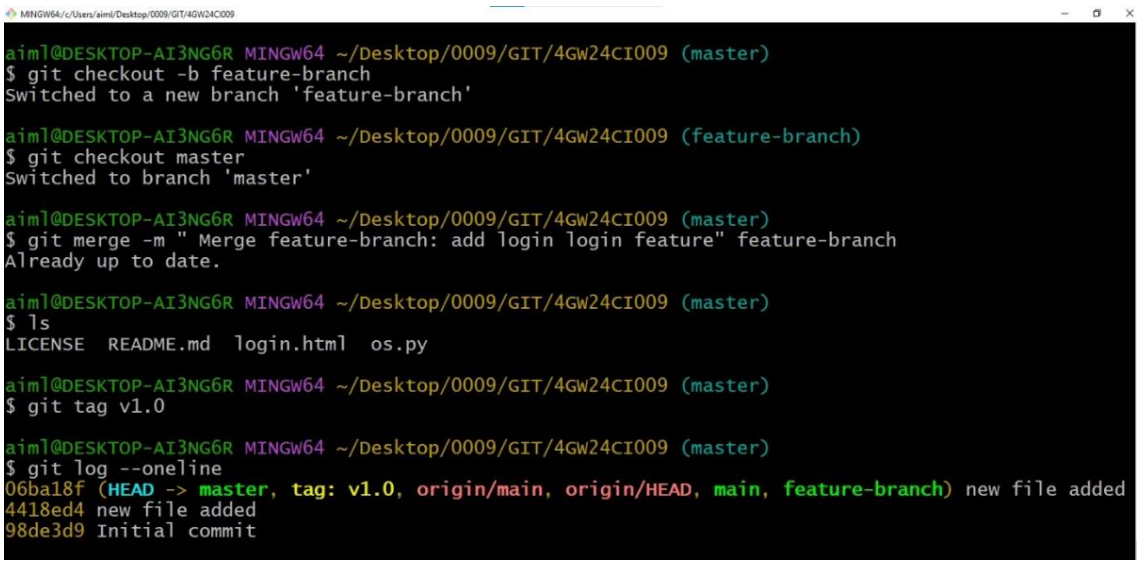
git merge -m "Merge feature-branch: add login feature" feature-branch



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.*

Using 'log' command we can see the commit with the tag we have used.

git log –oneline

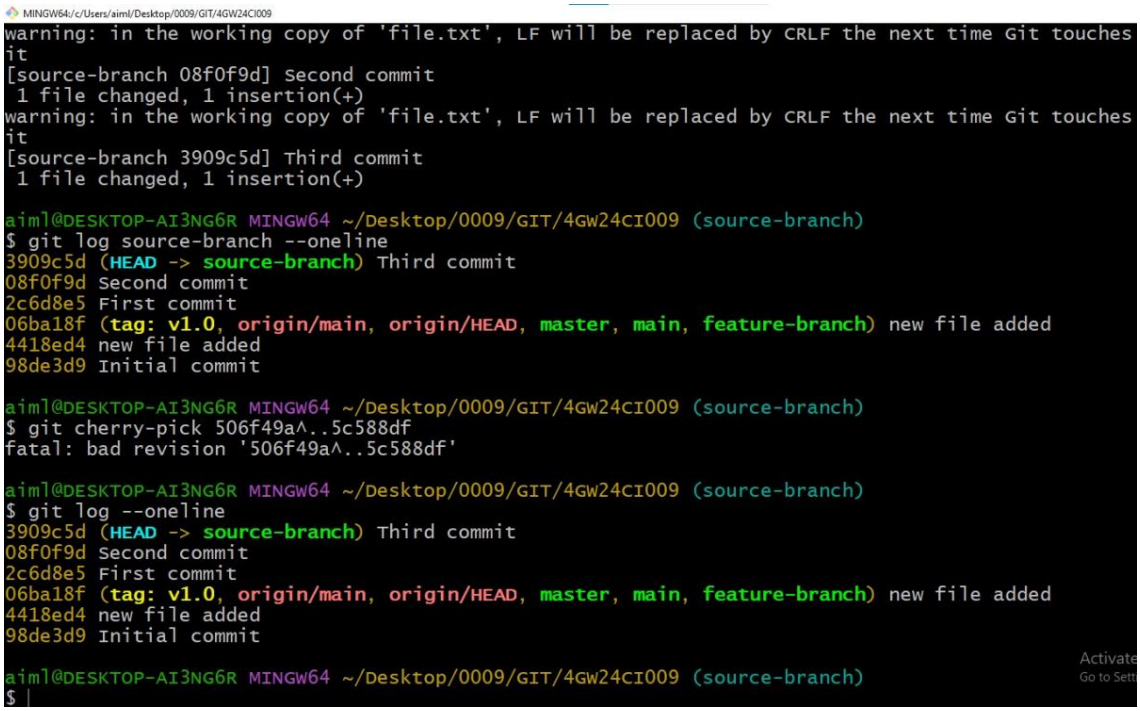


***Program 8**: Advanced Git Operations: Write the command to cherry-pick a range of commits from "source-branch" to the current branch*

git log source-branch --oneline -10

Using this command, we can see the commits with their id's. After this we use he 'cherry-pick' command. We give the range from the oldest id to the newest id.

git cherry-pick eea29d3^..68112df



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?*

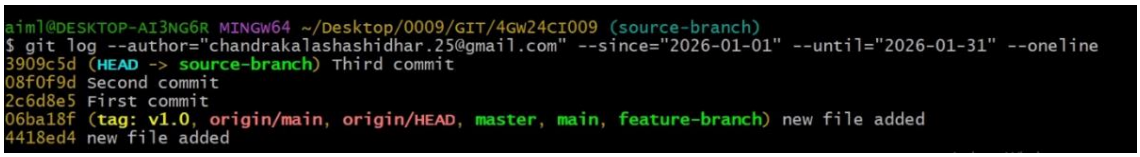
To view the history and all other details we use only one command that is the ‘show’ command

```
aiml@DESKTOP-AI3NG6R MINGW64 ~/Desktop/0009/GIT/4GW24CI009 (source-branch)
$ git show 3909c5d
commit 3909c5d4adc9e3296b8965e99c09eb30ef0201a6 (HEAD -> source-branch)
Author: chandrakalashashidhar25-maker <chandrakalashashidhar.25@gmail.com>
Date:   Wed Jan 7 12:40:19 2026 +0530

    Third commit

diff --git a/file.txt b/file.txt
index 1fefa18..6b8aee6 100644
--- a/file.txt
+++ b/file.txt
@@ -1,2 +1,3 @@
 First change
 Second change
+Third change
```

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."*

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"
```

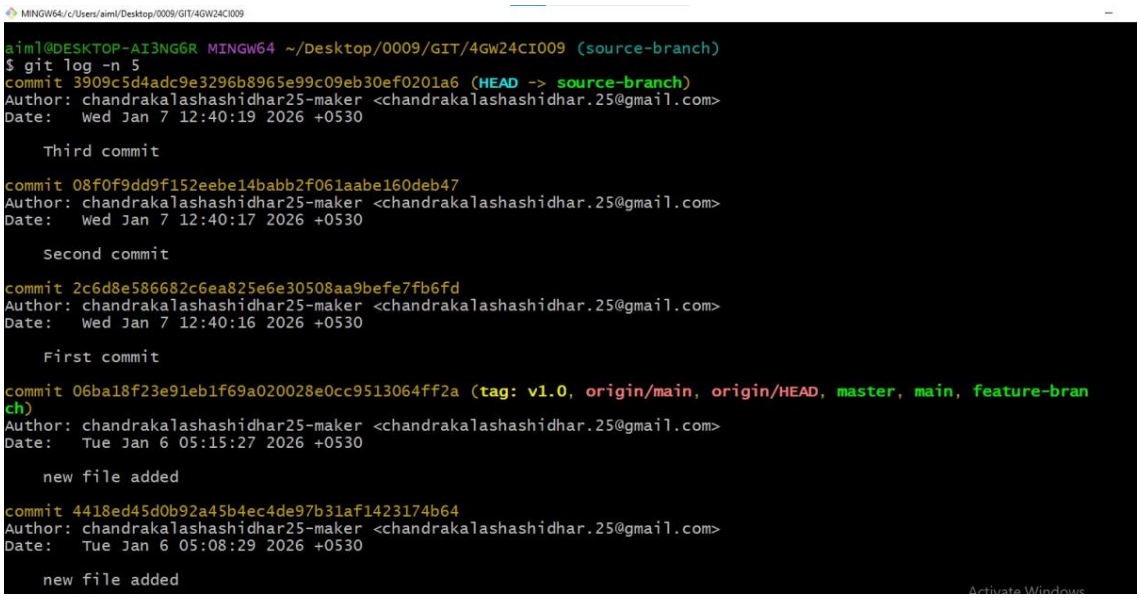


This command will display a list of commits made by the author "JohnDoe" that fall within the specified date range, from January 1, 2023, to December 31, 2023. Make sure to adjust the author’s

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

```
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.



Program 12*: Analyzing 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.