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

SIRI PATEL M

4GW24CI047

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

SI No	Contents	Page No.
1.	Setting Up and Basic Commands	1
2.	Creating and Managing Branches	1-2
3.	Creating and Managing Branches	3
4.	Collaboration and Remote Repositories	4
5.	Collaboration and Remote Repositories	4
6.	Collaboration and Remote Repositories	5
7.	Git Tags and Releases	6
8.	Advanced Git Operations	7
9.	Analyzing and Changing Git History	8
10.	Analyzing and Changing Git History	9
11.	Analyzing and Changing Git History	9
12.	Analyzing and Changing Git History	9-10
	Appendix	11

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.

```
-->First-Time Git Setup
git config --global user.name "username"
git config --global user.email "email"
git config --global init.defaultBranch main  ->to set main as default branch

-->basic commands
1.initialize a new git repo -> git init
2.create a new file -> touch filename

-->To Open File
i.Using vim command
vim filename
      i -> to insert line
      esc
      to save and exit                    to save without changes
      type wq+enter                       q!+enter
      Or :x+enter


ii.Using nano command
nano filename
      ctrl+s -> to save
      ctrl+x -> to exit

3.add it to the staging area -> git add filename Or git add.
4.commit changes --> git commit -m "message"
```

2.Creating and Managing branches

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

```
git branch branchname -> creates branch
git checkout branchname Or git switch branchname -> switch btw branches
git branch -> list all the branches

-->merging branches                                          ----------------------
if created along                                            | branch -> new branch|
git checkout sourcebranch (same like main branch)            ----------------------
git merge branch
if conflict -> open that file and resolve conflit
then merge again

if seperately created in same repo(created in Github and not present in local)
check if branch is there
try to merge, if not possible
git pull origin branch --allow-unrelated-histories
if conflict, resolve
git add .
git commit -m " " -> get message as "merged branch"
git push origin sourcebranch
```

```
Student@Student MINGW64 /d/ProjectManangement (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Student@Student MINGW64 /d/ProjectManangement (main)
$ git merge feature-branch
Auto-merging basics.txt
CONFLICT (content): Merge conflict in basics.txt
Automatic merge failed; fix conflicts and then commit the result.

Student@Student MINGW64 /d/ProjectManangement (main|MERGING)
$ nano basics.txt

Student@Student MINGW64 /d/ProjectManangement (main|MERGING)
$ git merge feature-branch
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.

Student@Student MINGW64 /d/ProjectManangement (main|MERGING)
$ git add .

Student@Student MINGW64 /d/ProjectManangement (main|MERGING)
$ git commit -m "merged feature-branch to main branch"
[main b9dad5e] merged feature-branch to main branch

Student@Student MINGW64 /d/ProjectManangement (main)
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 487 bytes | 243.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/                /ProjectManangement.git
   0f01955..b9dad5e  main -> main
```

3.Creating and Managing Branches

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

git stash list ->stash identifier/index, branch on which file is stashed, commit ID, commit message
git stash branch master stash{index} -> stores the changes into new branch master

```
Student@Student MINGW64 /d/ProjectManangement (master)
$ git stash
Saved working directory and index state WIP on master: e1271f5 stashed in new branch

Student@Student MINGW64 /d/ProjectManangement (master)
$ git stash list
stash@{0}: WIP on master: e1271f5 stashed in new branch

Student@Student MINGW64 /d/ProjectManangement (master)
$ nano file.txt

Student@Student MINGW64 /d/ProjectManangement (master)
$ git stash
Saved working directory and index state WIP on master: e1271f5 stashed in new branch

Student@Student MINGW64 /d/ProjectManangement (master)
$ git stash list
stash@{0}: WIP on master: e1271f5 stashed in new branch
stash@{1}: WIP on master: e1271f5 stashed in new branch

Student@Student MINGW64 /d/ProjectManangement (master)
$ git stash apply stash@{1}
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:   file.txt

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

Student@Student MINGW64 /d/ProjectManangement (master)
$ git stash list
stash@{0}: WIP on master: e1271f5 stashed in new branch
stash@{1}: WIP on master: e1271f5 stashed in new branch

Student@Student MINGW64 /d/ProjectManangement (master)
$ nano file.txt

Student@Student MINGW64 /d/ProjectManangement (master)
$ git stash -m "stash@{1} reapplied"
Saved working directory and index state On master: stash@{1} reapplied

Student@Student MINGW64 /d/ProjectManangement (master)
$ git stash list
stash@{0}: On master: stash@{1} reapplied
stash@{1}: WIP on master: e1271f5 stashed in new branch
stash@{2}: WIP on master: e1271f5 stashed in new branch

Student@Student MINGW64 /d/ProjectManangement (master)
$ git stash clear

Student@Student MINGW64 /d/ProjectManangement (master)
$ git stash list
```

4.Collaboration and Remote repository

Clone a remote Git repository to your local machine

```
Student@Student MINGW64 /d (master)
$ git clone https://github.com/                /mtd_DSA.git
Cloning into 'mtd_DSA'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
```

5.Collaboration and Remote repository

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

To get the latest changes from default branch

```
Student@Student MINGW64 /d/mtd_DSA (main)
$ git pull
Updating 62decca..555ebcf
Fast-forward
 text2.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 text2.txt

Student@Student MINGW64 /d/mtd_DSA (main)
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/                /mtd_DSA.git
  Push  URL: https://github.com/                /mtd_DSA.git
  HEAD branch: main
  Remote branch:
    main tracked
  Local branch configured for 'git pull':
    main merges with remote main
  Local ref configured for 'git push':
    main pushes to main (up to date)
```

To get the changes from newly created branch

```
Student@Student MINGW64 /d/mtd_DSA (main)
$ git checkout master
Switched to branch 'master'
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

Student@Student MINGW64 /d/mtd_DSA (master)
$ git pull
Updating 44f78ea..7038e55
Fast-forward
 file2.py | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 file2.py
```

To do the same with fetch command

i. To get changes from default branch to local

git fetch origin (default branch)

git merge

ii. To get changes from different branch to local

git fetch origin-> downloads changes from git repo

git checkout (default branch)

git merge (different branch)

6.Collaboration and Remote repository

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

```
Student@Student MINGW64 /d/mtd_DSA (main)
$ git rebase master
dropping 555ebcf54ac9d51597787721866b0944b346d6b4 Add text2.txt for testing git fetch commands -- patch
contents already upstream
dropping 74d2d23139f8df886ffd94a7798800859941e8b0 Initialize list and populate with range values -- patc
h contents already upstream
Successfully rebased and updated refs/heads/main.
```

```
Student@Student MINGW64 /d/mtd_DSA (master)
$ git fetch origin
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (1/1), 890 bytes | 55.00 KiB/s, done.
From https://github.com/                /mtd_DSA
   7038e55..920406e  master     -> origin/master

Student@Student MINGW64 /d/mtd_DSA (master)
$ git checkout main
Switched to branch 'main'
Your branch and 'origin/main' have diverged,
and have 2 and 5 different commits each, respectively.
  (use "git pull" if you want to integrate the remote branch with yours)
```

if conflict ->resolve conflict in the specified fil

if conflict -> resolve conflict in the specified file

git add <resolved-file>

git rebase --continue

7.Git tags and releases

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

```
Student@Student MINGW64 /d/GitHub (main)
$ nano text.txt

Student@Student MINGW64 /d/GitHub (main)
$ git add .

Student@Student MINGW64 /d/GitHub (main)
$ git commit -m "made changes"
[main d99f97b] made changes
 1 file changed, 3 insertions(+), 1 deletion(-)

Student@Student MINGW64 /d/GitHub (main)
$ git tag -a v1.0 d99f97b -m "Released v1.0"

Student@Student MINGW64 /d/GitHub (main)
$ git push origin v1.0
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 458 bytes | 114.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/siripatelm-stack/GitHub.git
 * [new tag]         v1.0 -> v1.0

Student@Student MINGW64 /d/GitHub (main)
$ git tag
v1.0

Student@Student MINGW64 /d/GitHub (main)
$ nano text1.txt

Student@Student MINGW64 /d/GitHub (main)
$ git tag v2.0

Student@Student MINGW64 /d/GitHub (main)
$ git add .

Student@Student MINGW64 /d/GitHub (main)
$ git commit -m "wrote some lines"
[main fb124fe] wrote some lines
 1 file changed, 1 insertion(+)

Student@Student MINGW64 /d/GitHub (main)
$ git push origin v2.0
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/siripatelm-stack/GitHub.git
 * [new tag]         v2.0 -> v2.0

Student@Student MINGW64 /d/GitHub (main)
$ git tag
v1.0
v2.0
```

git tag v1.0 -> Create lightweight tag (current commit)

git tag v1.0 abc1234 -> Create lightweight tag (specific commit)

git tag -a v1.0 -m "Release version 1.0" -> Create annotated tag (with message)

git tag -a v1.0 abc1234 -m "Release v1.0"-> Create tag from specific commit with message

git tag -> lists existing tags

8.Advanced git operations

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

Git cherry-pick is a Git command that allows you to **selectively apply individual commits from one branch to another branch**. Unlike merge or rebase, which integrate entire branches, cherry-pick lets you **hand-pick specific commits** and replay them on top of your current branch.
git cherry-pick <commit-hash> -> Cherry-pick a single commit
git cherry-pick commit1 commit2 commit3 -> Cherry-pick multiple specific commits

```
Student@Student MINGW64 /d/GitHub (main)
$ git log main --oneline
e8eccb5 (HEAD -> main, origin/main, origin/HEAD) cherry-pick
5434006 again something new made changes
fb124fe (master) wrote some lines
d99f97b (tag: v2.0, tag: v1.0) made changes
7d62d89 made changes
720f1c0 git tag
cf42a54 deleted some files
93d9005 deleted a file
0fa65eb Merge remote-tracking branch 'refs/remotes/origin/main' new java file added
257f1d8 Initialize list and populate with range values
8356ef7 Add text2.txt for testing git fetch commands
ad4a1bb Add Hello class with main method
7038e55 Create file2.py
74d2d23 Initialize list and populate with range values
44f78ea Add initial C program that prints 'Hello'
c326a92 Add print statement for 'Hello World'
555ebcf Add text2.txt for testing git fetch commands
62decca Add text1.txt with initial content
d61e124 deleted
9f6cce3 Add file2.txt for learning purposes
ae44844 Add learning DSA note to text.txt

Student@Student MINGW64 /d/GitHub (main)
$ git cherry-pick fb124fe
Auto-merging text1.txt
CONFLICT (content): Merge conflict in text1.txt
error: could not apply fb124fe... wrote some lines
hint: After resolving the conflicts, mark them with
hint: "git add/rm <pathspec>", then run
hint: "git cherry-pick --continue".
hint: You can instead skip this commit with "git cherry-pick --skip".
hint: To abort and get back to the state before "git cherry-pick",
hint: run "git cherry-pick --abort".
hint: Disable this message with "git config set advice.mergeConflict false"

Student@Student MINGW64 /d/GitHub (main|CHERRY-PICKING)
$ nano text1.txt

Student@Student MINGW64 /d/GitHub (main|CHERRY-PICKING)
$ git add .

Student@Student MINGW64 /d/GitHub (main|CHERRY-PICKING)
$ git commit -m "cherry-picking"
[main ccc329f] cherry-picking
 Date: Mon Dec 8 21:04:00 2025 +0530
 2 files changed, 3 insertions(+), 3 deletions(-)
 create mode 100644 text1.c

Student@Student MINGW64 /d/GitHub (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

Student@Student MINGW64 /d/GitHub (main)
$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 380 bytes | 76.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/siripatelm-stack/GitHub.git
   e8eccb5..ccc329f  main -> main
```


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?

git log main --oneline -> lists commit-hashes from latest to last

git cherry-pick<hash-commit> ->Pick any commit-hash and revisit the changes.

```
Student@Student MINGW64 /d/GitHub (main)
$ git log --graph --oneline --all
* ccc329f (HEAD -> main, origin/main, origin/HEAD) cherry-picking
* e8eccb5 cherry-pick
* 5434006 again something new made changes
* fb124fe (master) wrote some lines
* d99f97b (tag: v2.0, tag: v1.0) made changes
* 7d62d89 made changes
* 720f1c0 git tag
* cf42a54 deleted some files
| *   920406e (rm) Merge pull request #1 from      .           /master-1
| |\
| | * b3f2eba Add Hello.java with a simple greeting
| |/
|/|
* | 93d9005 deleted a file
* |   0fa65eb Merge remote-tracking branch 'refs/remotes/origin/main' new java file added
|\ \
| * | ad4a1bb Add Hello class with main method
| * | 74d2d23 Initialize list and populate with range values
| * | 555ebcf Add text2.txt for testing git fetch commands
* | | 257f1d8 Initialize list and populate with range values
* | | 8356ef7 Add text2.txt for testing git fetch commands
| |/
|/|
* | 7038e55 Create file2.py
* | 44f78ea Add initial C program that prints 'Hello'
* | c326a92 Add print statement for 'Hello World'
|/
* 62decca Add text1.txt with initial content
* d61e124 deleted
* 9f6cce3 Add file2.txt for learning purposes
* ae44844 Add learning DSA note to text.txt

Student@Student MINGW64 /d/GitHub (main)
$ git show 7d62d89
commit 7d62d8923b3a8353d6032aa7b70d7619d7bc7c37
Author:
Date:   Mon Dec 8 20:52:57 2025 +0530

    made changes

diff --git a/text1.txt b/text1.txt
index 5d72405..e05f637 100644
--- a/text1.txt
+++ b/text1.txt
@@ -1 +1 @@
-another file to fetch changes from cloned git repo.
+this is to create git tags
```

git log --graph --oneline --all -> shows the changes in graph

git show<commit-hash> -> List the changes made in specific commit-hash

git log -> to see commit history

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."

```
Student@Student MINGW64 /d/GitHub (main)
$ git log --all --author="                " --after="2025-01-01" --before="2025-12-31" --onel
ine
ccc329f (HEAD -> main, origin/main, origin/HEAD) cherry-picking
e8eccb5 cherry-pick
5434006 again something new made changes
fb124fe (master) wrote some lines
d99f97b (tag: v2.0, tag: v1.0) made changes
7d62d89 made changes
720f1c0 git tag
```

git log --all --author=”JohnDoe” --after=”” --before=”” --oneline -> lists the history with commit-hash and commit message

git log --pretty=format:"%ad - %an: %s" --after="2023-01-01" --before="2023-12-31" --author="JohnDoe" -> lists the all the commit history in format

“day month date time year +timezone author :commit message”

```
Student@Student MINGW64 /d/GitHub (main)
$ git log -5 --oneline
6947eed (HEAD -> main, origin/main, origin/HEAD) Revert "cherry-picking"
ccc329f cherry-picking
e8eccb5 cherry-pick
5434006 again something new made changes
fb124fe (master) wrote some lines
```

git log -5 -> to see last 5 commit history

11.Analysing and Changing Git History

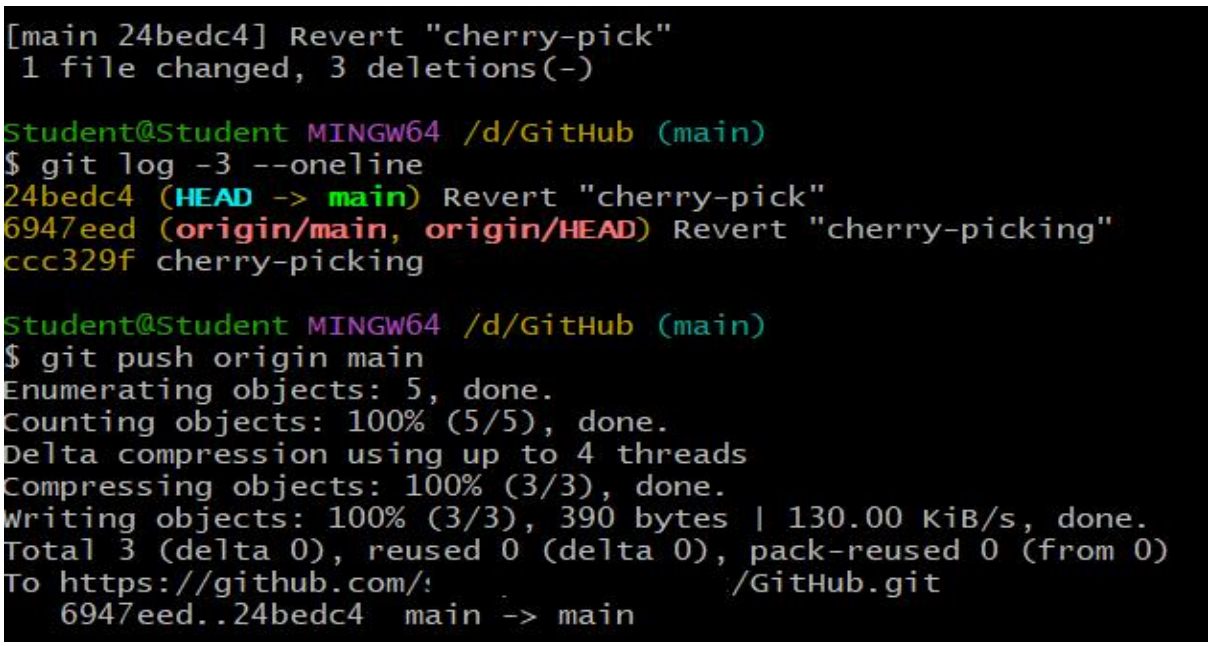
Write the command to display the last five commits in the repository's history.

git log -5 --oneline -> lists commit history in oneline

12.Analysing and Changing Git History

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

git revert <commit hash> -> reverts changes from particular commit


[main 24bedc4] Revert "cherry-pick"
1 file changed, 3 deletions(-)
Student@Student MINGW64 /d/GitHub (main)
$ git log -3 --oneline
24bedc4 (HEAD -> main) Revert "cherry-pick"
6947eed (origin/main, origin/HEAD) Revert "cherry-picking"
ccc329f cherry-picking
Student@Student MINGW64 /d/GitHub (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 390 bytes | 130.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/ /GitHub.git
6947eed..24bedc4 main -> main

Repo link-

https://github.com/siripatelm-stack/4GW24CI047.git