

# **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“Jnana Sangama”, Machhe, Belagavi, Karnataka-590018**



Lab Experiment Record

## **Project Management with Git [BCSL358C]**

*Submitted in partial fulfillment towards AEC of 3<sup>rd</sup> semester of*

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

Submitted by

**INAAM HANIYA YOUSUF  
[4GW24CI015]**



**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)**

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		
<b>Course objectives:</b>			
<ul style="list-style-type: none"><li>• To familiar with basic command of Git</li><li>• To create and manage branches</li><li>• To understand how to collaborate and work with Remote Repositories</li><li>• To familiar with version controlling commands</li></ul>			
SLNO	Experiments		
1	<b>Setting Up and Basic Commands</b> 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	<b>Creating and Managing Branches</b> Create a new branch named "feature-branch." Switch to the "master" branch. Merge the "feature-branch" into "master."		
3	<b>Creating and Managing Branches</b> Write the commands to stash your changes, switch branches, and then apply the stashed changes.		
4	<b>Collaboration and Remote Repositories</b> Clone a remote Git repository to your local machine.		
5	<b>Collaboration and Remote Repositories</b> Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch.		
6	<b>Collaboration and Remote Repositories</b> Write the command to merge "feature-branch" into "master" while providing a custom commit message for the merge.		
7	<b>Git Tags and Releases</b> Write the command to create a lightweight Git tag named "v1.0" for a commit in your local repository.		
8	<b>Advanced Git Operations</b>		

	Write the command to cherry-pick a range of commits from "source-branch" to the current branch.
9	<b>Analysing and Changing Git History</b>  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	<b>Analysing and Changing Git History</b>  Write the command to list all commits made by the author "JohnDoe" between "2023-01-01" and "2023-12-31."
11	<b>Analysing and Changing Git History</b>  Write the command to display the last five commits in the repository's history.
12	<b>Analysing and Changing Git History</b>  Write the command to undo the changes introduced by the commit with the ID "abc123".

## 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.

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ mkdir git1

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ git init
Reinitialized existing Git repository in C:/Users/4gw19/OneDrive/Desktop/git-lab-clean/.git/

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ touch file1.txt

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ git add file1.txt

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ git commit -m"initial commit"
[master (root-commit) f0de87d] initial commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 file1.txt
```

A new Git repository was successfully initialized and the file was committed.

Here mkdir is used to make a new directory i.e. a folder in the system.  
We use commands git init, git add, and git commit.

## Experiment 2.

### Creating and Managing Branches:

Create a new branch named "feature-branch."

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

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ git branch feature-branch

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ git checkout master
Already on 'master'

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ git checkout feature-branch
Switched to branch 'feature-branch'

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git switch master
Switched to branch 'master'

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ git merge feature-branch
Already up to date.
```

Changes were stashed and successfully applied on another branch.

Here we have used the command **checkout** to switch branches, we can also use the command **switch** to do the same.

**Merge** is used to merge the two existing branches.

## Experiment 3.

### Creating and Managing Branches:

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

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ git stash
Saved working directory and index state WIP on master: f0de87d initial commit
```

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git stash apply
On branch feature-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:   file1.txt

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

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git status
On branch feature-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:   file1.txt

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

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git add file1.txt

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git commit -m"a"
[feature-branch 55d4912] a
 1 file changed, 1 insertion(+)

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git status
On branch feature-branch
nothing to commit, working tree clean

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git stash apply
On branch feature-branch
nothing to commit, working tree clean
```

Changes were stashed and successfully applied on another branch.  
Here for **stash apply** the modified file had to first be committed.  
We have used the commands **stash** and **stash apply**.

## Experiment 4.

### Collaboration and Remote Repositories:

Clone a remote Git repository to your local machine.

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git clone https://github.com/ina528/4GW24CI015.git
Cloning into '4GW24CI015'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

The remote repository was successfully cloned.

Here the only command used is **clone** followed by the repository address link from GitHub.

## Experiment 5.

### Collaboration and Remote Repositories:

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

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git fetch origin
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git remote -v

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git remote add origin https://github.com/ina528/4GW24CI015.git

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git remote -v
origin https://github.com/ina528/4GW24CI015.git (fetch)
origin https://github.com/ina528/4GW24CI015.git (push)

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git fetch origin
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 869 bytes | 57.00 KiB/s, done.
From https://github.com/ina528/4GW24CI015
* [new branch]      main      -> origin/main
```

Local branch was rebased with the updated remote branch.  
Initially, fetch failed because the remote repository was not configured.  
After adding the remote repository, fetch and rebase were executed successfully

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ git rebase origin/main
Successfully rebased and updated refs/heads/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.

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/gitlab (master)
$ git checkout master
Already on 'master'
Your branch is up to date with 'origin/master'.

4gw19@LAPTOP-NGUM35FU MINGW64 ~/gitlab (master)
$ git merge feature-branch -m"merged into master"
Already up to date.
```

Branches were merged with a custom commit message.  
Here we use the command **merge**.

## 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.

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git tag v1.0

4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (feature-branch)
$ git tag
v1.0
```

Lightweight tag v1.0 was created successfully.  
Here the **tag** command is used. It can be used both to create a tag and list the existing tags.

## Experiment 8.

### Advanced Git Operations:

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

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/gitlab (master)
$ git cherry-pick 38e795a2849dac95b8a6da5966bfe7fd28ebface
Auto-merging file.txt
```

Specified commits were cherry-picked successfully.  
Here the command **cherry-pick** is used followed by the commit id.

## 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.

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (other-branch)
$ git show cc9d3a185f6eb89a0cfac1b02c5f51daf7e2b85b
commit cc9d3a185f6eb89a0cfac1b02c5f51daf7e2b85b
Author: ina528 <forstudiesalwayz@gmail.com>
Date:   Tue Jan 6 10:37:04 2026 +0530

    commit 1

diff --git a/file1.txt b/file1.txt
index 2cfe96a..73a45d6 100644
--- a/file1.txt
+++ b/file1.txt
@@ -1,2 @@
- hicommit
+ hicommit commit 1
+commit
```

Commit details including author, date, and message were displayed.  
Here the command show is used followed by the commit id of the particular commit that we want to see.  
We can see the author name that is the GitHub username followed by the user Gmail id, the date and the commit message.

## 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."

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (other-branch)
$ git log --author="ina528" --since="2026-01-01" --until="2026-01-06"
commit a5f8d997fbddfdf964cfd176ddeddd19571dbc54 (HEAD -> other-branch)
Author: ina528 <forstudiesalwayz@gmail.com>
Date: Tue Jan 6 10:38:55 2026 +0530

    commit 5

commit 9ccf0bbe263bb114b17b06a3838249ccf6f6aa6e
Author: ina528 <forstudiesalwayz@gmail.com>
Date: Tue Jan 6 10:38:37 2026 +0530

    commit 4

commit fb80e9a12b6c064b6e0663a0e85c57b5a1c7805a
Author: ina528 <forstudiesalwayz@gmail.com>
Date: Tue Jan 6 10:38:14 2026 +0530

    commit 3

commit 735b44c8af96771f4accb67ff50ce88684ef5e54
Author: ina528 <forstudiesalwayz@gmail.com>
Date: Tue Jan 6 10:37:40 2026 +0530

    commit 2

commit cc9d3a185f6eb89a0cfac1b02c5f51daf7e2b85b
Author: ina528 <forstudiesalwayz@gmail.com>
Date: Tue Jan 6 10:37:04 2026 +0530

    commit 1

commit fcefe31253b94bdc57d872427d52142ebd929602
Author: ina528 <forstudiesalwayz@gmail.com>
Date: Tue Jan 6 10:36:14 2026 +0530

    commit 1

commit 55d4912254e3e92fc35ba8c4d8d26f804cdc27b8 (tag: v1.0, origin/feature-branch, feature-branch)
```

Commits by the specified author during the given period are displayed.  
Here the **log --author="author name" --since="start date" --until="end date"**  
This command lists all the commits done between the two dates by the author.

## Experiment 11

### Analysing and Changing Git History

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

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ git log -5
```

```
commit a5f8d997fbddfdf964cfd176ddeddd19571dbc54 (HEAD -> other-branch)
commit a5f8d997fbddfdf964cfd176ddeddd19571dbc54 (HEAD -> other-branch)
Author: ina528 <forstudiesalwayz@gmail.com>
Date: Tue Jan 6 10:38:55 2026 +0530

    commit 5

commit 9ccf0bbe263bb114b17b06a3838249ccf6f6aa6e
Author: ina528 <forstudiesalwayz@gmail.com>
Date: Tue Jan 6 10:38:37 2026 +0530

    commit 4

commit fb80e9a12b6c064b6e0663a0e85c57b5a1c7805a
Author: ina528 <forstudiesalwayz@gmail.com>
Date: Tue Jan 6 10:38:14 2026 +0530

    commit 3

commit 735b44c8af96771f4accb67ff50ce88684ef5e54
Author: ina528 <forstudiesalwayz@gmail.com>
Date: Tue Jan 6 10:37:40 2026 +0530

    commit 2

commit cc9d3a185f6eb89a0cfac1b02c5f51daf7e2b85b
Author: ina528 <forstudiesalwayz@gmail.com>
Date: Tue Jan 6 10:37:04 2026 +0530

    commit 1
```

Last five commits were displayed successfully.  
Here we use the command **log -5** which lists out the commits.

## Experiment 12.

### Analysing and Changing Git History

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

```
4gw19@LAPTOP-NGUM35FU MINGW64 ~/OneDrive/Desktop/git-lab-clean (master)
$ git revert e1ec249e4834fe434412c14fc8e15e5b411cdc21|
```

```
[other-branch fbebbc4] Revert "commit 4"
1 file changed, 1 insertion(+)
```

The specified commit was successfully reverted.  
Here we use the **revert** command foll by the commit id