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
VARSHA SURESH
4GW24CI060

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

Contents

Experiment 1: Setting Up and Basic Commands

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.

- Step 1: Configure Git User Information
  git config --global user.name "SnehagangaNS"
  git config --global user.email "snehaganganadumane@gmail.com"
  This command sets the user name and email for Git.

- Step 2: Initialize Git Repository
  git init
  This command creates a new Git repository in the current folder.

- Step 3: Create a File
  touch test.txt
  This command creates a new empty file named test.txt.

- Step 4: Check Status
  git status
  This command shows the current status of the repository.

- Step 5: Add Files
  git add .
  This command adds all files to the staging area.

- Step 6: Commit the Changes
  git commit -m "Initial commit: added file1.txt"
  This command saves the changes to the repository

This experiment initializes a new Git repository, adds a file to the staging area, and commits the changes with an appropriate commit message.

```
surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG
$ git init
Initialized empty Git repository in C:/Varsha/Crack
athon_GDG/.git/

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG
 (main)
$ git status
On branch main

No commits yet

Untracked files:
   (use "git add <file>..." to include in what will
be committed)
        .gitignore
        README.md
        app.py
        assets.py
        create_dataset.py
        gemini_ai.py
        hand_landmarks.csv
        hand_tracker.py
        label_encoder.pkl
        logo.png
        packages.txt
        requirements.txt
        sign_language_model.h5
        spelling_assets/
        styles.css
        train_model.py
        words_alpha.txt

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

```
surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG
 (main)
$ git add .

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG
 (main)
$ git status
On branch main

No commits yet

Changes to be committed:
   (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   README.md
        new file:   app.py
        new file:   assets.py
        new file:   create_dataset.py
        new file:   gemini_ai.py
        new file:   hand_landmarks.csv
        new file:   hand_tracker.py
        new file:   label_encoder.pkl
        new file:   logo.png
        new file:   packages.txt
        new file:   requirements.txt
```

Experiment 2: Creating and Managing Branches

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

Branching and merging in Git enable developers to work in isolation from the main codebase, facilitate parallel development, and maintain a stable, high-quality main branch.

- Step 1: Create a New Branch
  git branch feature-branch
  This command creates a new branch called feature-branch.

- Step 2: Switch to the Feature Branch
  git checkout feature-branch
  This command switches from the master branch to the feature branch.

- Step 3: Make Changes in Feature Branch
  git add .
  git commit -m "Added changes in feature-branch"
  Changes are made in the feature branch.
  The updated files are added and committed.

- Step 4: Switch Back to Master Branch
  git checkout master
  This command switches back to the main (master) branch.

- Step 5: Merge Feature Branch into Master
  git merge feature-branch
  This command merges the changes from feature-branch into master.

```
surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (main)
$ git branch feature-branch

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG
 (main)
$ git branch
  feature-branch
* main

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG
 (main)
$ git checkout feature-branch
Switched to branch 'feature-branch'

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG
 (feature-branch)
$ git add words_alpha.txt
warning: in the working copy of 'words_alpha.txt',
LF will be replaced by CRLF the next time Git touch
es it

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG
 (feature-branch)
$ git add words_alpha.txt

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG
 (feature-branch)
$ git commit -m "Updated this text fild in deature-
branch"
[feature-branch 0c6fd13] Updated this text fild in
deature-branch
 1 file changed, 1 insertion(+)
```

```
surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG
 (feature-branch)
$ git checkout main
Switched to branch 'main'

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG
 (main)
$ git merge feature-branch
Updating 8af7d08..0c6fd13
Fast-forward
 words_alpha.txt | 1 +
 1 file changed, 1 insertion(+)

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG
 (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will
be committed)
        words_alpha.txt.bak

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

This experiment demonstrates how to create a new branch, switch between branches, and merge a feature branch into the main branch.

Experiment 3: Creating and Managing Branches

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

- Step 4: Create and Switch to a Feature Branch
  git branch feature-branch
  git checkout feature-branch
  Creates a new branch called feature-branch.
  Switches from the master branch to the feature branch.

- Step 6: Switch Back to Master Branch
  git checkout master
  Returns to the master branch.
  The file test.txt still exists in this branch.

- Step 7: Stash the Current Changes
  git stash
  Temporarily saves uncommitted changes.

- Step 8: Switch to Feature Branch and Apply Stash
  git checkout feature-branch
  git stash apply
  Switches back to the feature branch.
  Applies the stashed changes to this branch.

```
surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (ma
in)
$ git stash
warning: in the working copy of 'README.md', LF will be
 replaced by CRLF the next time Git touches it
Saved working directory and index state WIP on main: Oc
6fd13 Updated this text fild in deature-branch

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (ma
in)
$ git stash
No local changes to save

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (ma
in)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be c
ommitted)
        words_alpha.txt.bak

nothing added to commit but untracked files present (us
e "git add" to track)

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (ma
in)
$ git checkout feature-branch
Switched to branch 'feature-branch'

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (fe
ature-branch)
$ git stash apply
On branch feature-branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be commi
tted)
  (use "git restore <file>..." to discard changes in wo
rking directory)
        modified:   README.md
```

Experiment 4: Collaboration and Remote Repositories

Clone a remote Git repository to your local machine.

We clone a repository to get a complete local copy of a project (including its full history) onto our computer, allowing us to work on it offline, make changes, fix issues, add features, and then sync those updates back to the central remote repository.

- Step 1: Clone a Remote Repository
  git clone <repository-url>

```
surit@Dell-VarshaS MINGW64 /c/Varsha/GitLabPractice (feature-branch)
$ git clone https://github.com/SnehagangaNS/project-demo.git
Cloning into 'project-demo'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 14 (delta 1), reused 13 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (14/14), 5.07 KiB | 1.27 MiB/s, done.
Resolving deltas: 100% (1/1), 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.

Update local branch with remote changes using rebase. Downloads new changes from the remote without merging. Applies local commits on top of the updated remote branch, keeping history clean.

- Step 1: Fetch Latest Changes from Remote Repository
  git fetch origin
- Step 2: Rebase Feature Branch onto Remote Master
  git rebase origin/main

```
surit@Dell-VarshaS MINGW64 /c/Varsha/GitLabPractice (main)
$ git branch
  desbranch
  feature-branch
* main

surit@Dell-VarshaS MINGW64 /c/Varsha/GitLabPractice (main)
$ git fetch origin

surit@Dell-VarshaS MINGW64 /c/Varsha/GitLabPractice (main)
$ git rebase origin/main
error: cannot rebase: Your index contains uncommitted changes.
error: Please commit or stash them.

surit@Dell-VarshaS MINGW64 /c/Varsha/GitLabPractice (main)
$ git stash
Saved working directory and index state WIP on main: 4ab2dfd Initial commit - adding an empty text file
warning: unable to rmdir 'project-demo': Directory not empty

surit@Dell-VarshaS MINGW64 /c/Varsha/GitLabPractice (main)
$ git rebase origin/main
Successfully rebased and updated refs/heads/main.

surit@Dell-VarshaS MINGW64 /c/Varsha/GitLabPractice (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        project-demo/

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

This experiment fetches the latest changes from the remote repository and rebases the local branch onto the updated remote branch to maintain a clean history.

Experiment 6: Collaboration and Remote Repositories

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

- Step 1: Switch to Master Branch
  git checkout master
  This command switches the current branch to master.

- Step 2: Merge Feature Branch into Master
  git merge source-branch -m "Merged source-branch into master"
  This command merges feature-branch into the master branch.

```
surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (fe
ature-branch)
$ git branch
* feature-branch
  main

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (feature
-branch)
$ git checkout main
M       README.md
Switched to branch 'main'

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (main)
$ git merge feature-branch -m "merge feature-branch into mas
ter"
Already up to date.

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (main)
$ git log --oneline --graph
* 0c6fd13 (HEAD -> main, feature-branch) Updated this text f
ild in deature-branch
* 8af7d08 Initial commit - Added the files

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (main)
$ git branch
  feature-branch
* main
```

This experiment merges a feature branch into the main branch while providing a custom merge commit message.

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.

Command:
git tag v1.0

```
surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (main)
$ git log --oneline
0c6fd13 (HEAD -> main, feature-branch) Updated t
his text fild in deature-branch
8af7d08 Initial commit - Added the files

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_
GDG (main)
$ git tag v1.0

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_
GDG (main)
$ git tag
v1.0
```

This experiment creates a lightweight Git tag named v1.0 that points to a specific commit in the repository.

Experiment 8: Advanced Git Operations

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

Apply specific commits from another branch. Selectively applies a range of commits without merging the entire branch.

First get the commit id, use this for cherry-pick.
- Step 1: View Commits in Source Branch
  git log source-branch --oneline
  Displays the list of commits made in source-branch.
  Shows commit IDs in a short format.
  These commit IDs are needed for cherry-picking.

- Step 2: Cherry-Pick a Range of Commits
  git cherry-pick <start-commit>^..<end-commit>
  This command copies a specific range of commits from source-branch.
  The selected commits are applied to the master branch.
  Only chosen commits are merged, not the entire branch.

```
surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (main)
$ git checkout feature-branch
M       README.md
Switched to branch 'feature-branch'

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (feature-branch)
$ git add .
warning: in the working copy of 'words_alpha.txt.bak', LF will be replaced by CRLF the next time Git touches it

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (feature-branch)
$ git add .

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (feature-branch)
$ git commit -m "this is a message"
[feature-branch e6245ea] this is a message
 2 files changed, 370106 insertions(+)
 create mode 100644 words_alpha.txt.bak

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (feature-branch)
$ git checkout main
Switched to branch 'main'

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (main)
$ git log feature-branch --oneline
e6245ea (feature-branch) this is a message
0c6fd13 (HEAD -> main, tag: v1.0) Updated this text fild in deature-branch
8af7d08 Initial commit - Added the files

surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (main)
$ git cherry-pick e6245ea
[main f83ae50] this is a message
 Date: Tue Jan 6 09:28:43 2026 +0530
 2 files changed, 370106 insertions(+)
 create mode 100644 words_alpha.txt.bak
```

This experiment applies a selected range of commits from a source branch onto the current branch using cherry-pick.

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?

- Step: Display Commit Information
  git show <commit-id>
  This command displays detailed information about a specific commit.
  It shows the commit author, date, commit message, and the changes made.

```
$ git show
commit 0c6fd13fe3ee5e7fed678e07d27ec40e103b8855 (HEAD -> main, tag: v1.0, feature-branch)
Author: varsha-sureshh <varsha.suresh.mys@gmail.com>
Date:   Tue Jan 6 05:39:09 2026 +0530

    Updated this text fild in deature-branch

diff --git a/words_alpha.txt b/words_alpha.txt
index 997a7a9..4307056 100644
--- a/words_alpha.txt
+++ b/words_alpha.txt
@@ -370103,3 +370103,4 @@ zwinglianist
 zwitter
 zwitterion
 zwitterionic
+// This line was added in feature-branch.
```

This shows the details made 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."

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

```
surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (main)
$  git log --author="Dell-VarshaS" --since="2025-01-01" --until="2025-12-31"
```

This experiment displays all commits made by a specific author within a given date range.

Experiment 11: Analysing and Changing Git History

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

Command -
*   git log -5

```
surit@Dell-VarshaS MINGW64 /c/Varsha/Crackathon_GDG (main)
$ git log -5
commit f83ae50c7dfa7429a57fad72db720c3ba6f9588e (HEAD -> main)
Author: varsha-sureshh <varsha.suresh.mys@gmail.com>
Date:   Tue Jan 6 09:28:43 2026 +0530

    this is a message

commit 0c6fd13fe3ee5e7fed678e07d27ec40e103b8855 (tag: v1.0)
Author: varsha-sureshh <varsha.suresh.mys@gmail.com>
Date:   Tue Jan 6 05:39:09 2026 +0530

    Updated this text fild in deature-branch

commit 8af7d08026c76d2c1e8437fc01f1072b817bb9a8
Author: varsha-sureshh <varsha.suresh.mys@gmail.com>
Date:   Tue Jan 6 05:30:00 2026 +0530

    Initial commit - Added the files
```

This experiment displays the most recent five commits in the repository's history

Experiment 12: Analysing and Changing Git History

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

Command -
- git revert <abc123>

This experiment undoes the changes introduced by a specific commit by creating a new commit that reverses those changes.

```
[main cecdd89] Revert "Updated this text fild in deature-branch"
 1 file changed, 1 deletion(-)
```

Appendix

Below is the link to my GitHub repository where I have added the lab programs from different subjects.
https://github.com/varsha-sureshh/GithubLab