EXPERIMENT No. 01: Setting Up and Basic Commands

1.1 Objective

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

1.2 System Configuration

Windows 10, Linux,Mac

1.3 Pre-Requisite

Install Git bash and make required settings.

1.4 Introduction

git init

This command is used to initialize a new Git repository in the current directory. It sets up the necessary data structures and files that Git needs to start tracking changes to your project.

git clone ‘remote repository link’

This command is used to create a copy of an existing Git repository from a remote location (in this case, from the specified URL) to your local machine. It not only copies the files but also sets up a connection to the original repository so you can pull in updates later.

git add .

This command is used to stage all changes in the current directory for the next commit. Staging is the process of preparing files to be included in the next commit. The dot (.)

indicates that all changes, including new files, modified files, and deleted files, should be staged.

git commit -m "<message>"

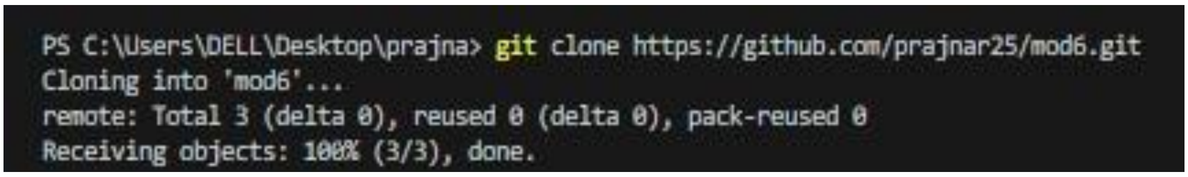
This command is used to record the changes that have been staged (using git add) in the repository. The -m flag allows you to include a short message that describes the changes made in this commit. It's good practice to write clear and concise commit messages that explain the purpose of the changes.

git push

This command is used to upload local repository content to a remote repository. In the context of GitHub, it typically sends committed changes from your local repository to the remote repository on GitHub. This allows others to see the changes you've made and collaborate with you. Depending on your Git configuration, you may need to specify the remote repository and branch name, but if you've cloned a repository, Git usually sets up the default remote and branch for you.

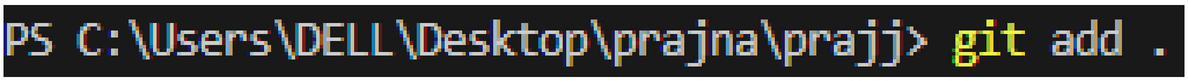
1.5 Procedure and Results

git clone ‘remote repository link’



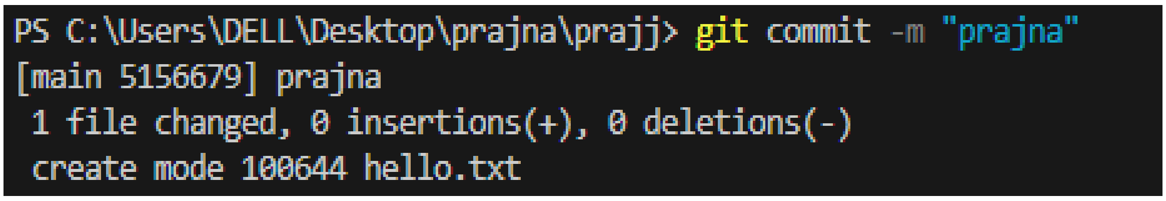
```
PS C:\Users\DELL\Desktop\prajna> git clone https://github.com/prajnar25/mod6.git
Cloning into 'mod6'...
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

git add .



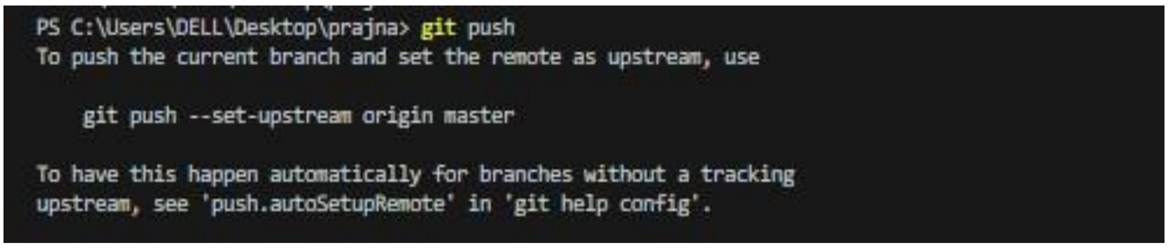
```
PS C:\Users\DELL\Desktop\prajna\prajj> git add .
```

git commit -m "<message>"



```
PS C:\Users\DELL\Desktop\prajna\prajj> git commit -m "prajna"
[main 5156679] prajna
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hello.txt
```

git push



```
PS C:\Users\DELL\Desktop\prajna> git push
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin master

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
```

EXPERIMENT No. 02: Creating and Managing Branches

2.1Objective

a) Create a new branch named “feature-branch”. Switch to the “master” branch. Merge the “feature-branch” into “master”.

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

2.2 System Configuration

Windows 10, Linux, Mac

2.3 Pre-Requisite

1. Git is installed on your system.
2. You have a Git repository initialized and have some changes made to the files.
3. You have at least two branches: master/main and feature-branch.

2.4 Introduction

. git branch -b feature-branch

This command creates a new branch named "feature-branch" but doesn't switch to it.

git checkout master/main

This command switches to the "master" branch.

git merge feature-branch

This command merges changes from "feature-branch" into the "master" branch git stash:

Stash your changes

git checkout feature-branch
Switch branches (example: from "master" to "feature-branch")

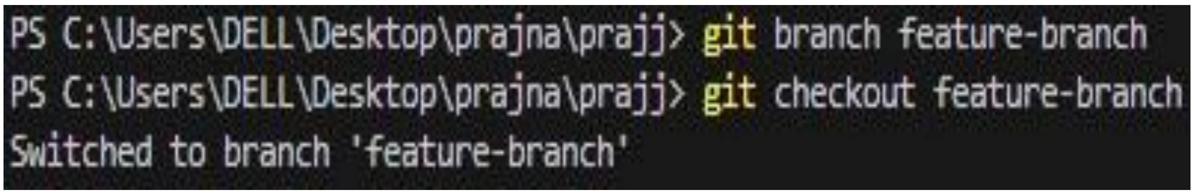
git stash apply

Apply the stashed changes

2.5 Procedure and Results

git branch feature-branch

git checkout feature-branch



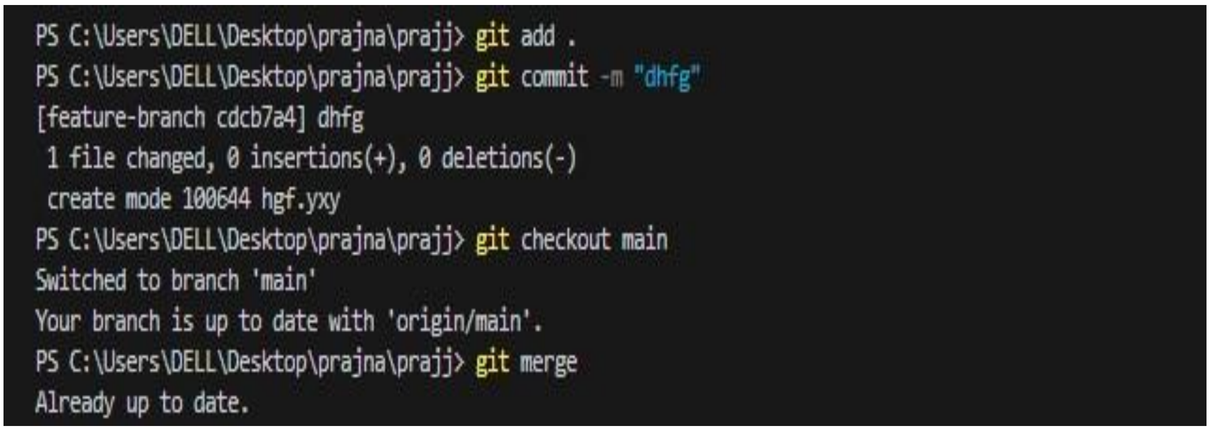
```
PS C:\Users\DELL\Desktop\prajna\prajj> git branch feature-branch
PS C:\Users\DELL\Desktop\prajna\prajj> git checkout feature-branch
Switched to branch 'feature-branch'
```

git add .

git commit -m “<message>”

git checkout master git

merge

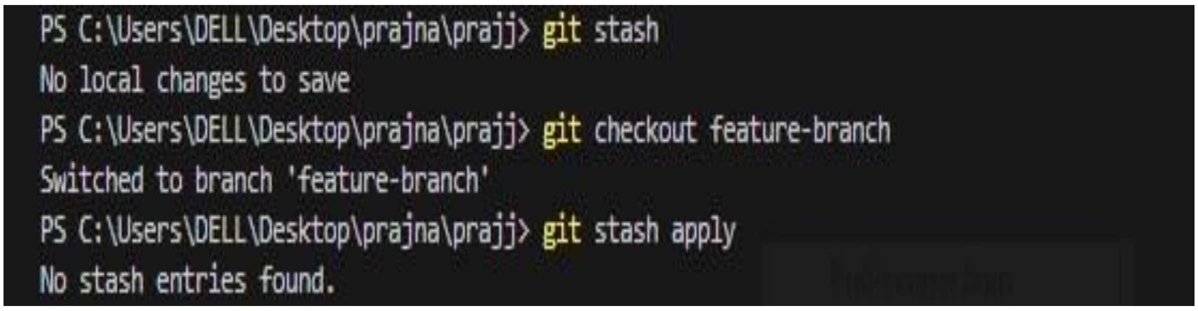


```
PS C:\Users\DELL\Desktop\prajna\prajj> git add .
PS C:\Users\DELL\Desktop\prajna\prajj> git commit -m "dhfg"
[feature-branch cdcb7a4] dhfg
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hgf.yxy
PS C:\Users\DELL\Desktop\prajna\prajj> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\DELL\Desktop\prajna\prajj> git merge
Already up to date.
```

git stash

git checkout feature-branch git

stash apply



```
PS C:\Users\DELL\Desktop\prajna\prajj> git stash
No local changes to save
PS C:\Users\DELL\Desktop\prajna\prajj> git checkout feature-branch
Switched to branch 'feature-branch'
PS C:\Users\DELL\Desktop\prajna\prajj> git stash apply
No stash entries found.
```

EXPERIMENT No.03: Collaboration and Remote Repositories

3.1 Objectives:

a) Clone a remote Git repository to your local machine.

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

c) Write a command to merge “feature-branch” into “master” while providing a custom commit message for the merge.

3.2 System Configuration:

Windows 10

3.3 Pre-Requisite:

4 Git bash should be installed

5 Visual studio code editor should be installed

3.4 Introduction:

a) Clone a remote Git repository to your local machine:

```
git clone <repository_url>
```

This command clones a remote Git repository onto your local machine, creating a new directory with the same name as the repository.

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

```
git  checkout  -b  feature-branch
git fetch origin
git rebase origin/<branch_name>
```

The first command (git fetch origin) fetches the latest changes from the remote repository named "origin" without merging them into your local branches. The second command (git rebase origin/<branch_name>) rebases your current local branch onto the updated remote branch, integrating the latest changes from the remote repository while maintaining the commit history.

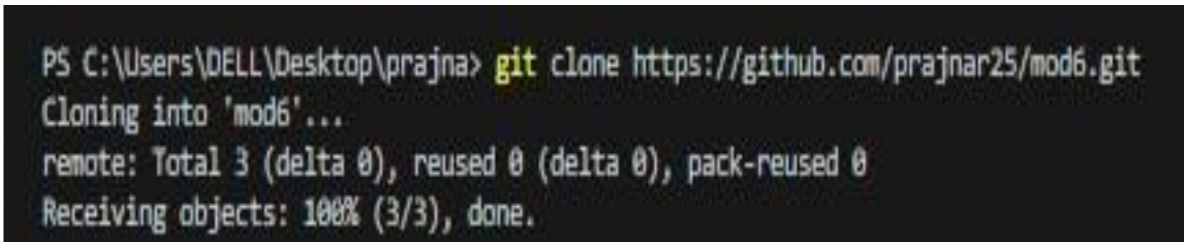
c) Write a command to merge “feature-branch” into “master” while providing a custom commit message for the merge:

```
git checkout master
git merge --no-ff feature-branch -m "Custom commit message"
```

The first command (git checkout master) switches to the "master" branch. The second command (git merge --no-ff feature-branch -m "Custom commit message") merges the changes from the "feature-branch" into the "master" branch, creating a merge commit with the provided custom commit message. The --no-ff option ensures that a merge commit is always created, even if the merge could be performed with a fast-forward.

3.5 Procedure and Results:

```
git clone <repository_url>
```

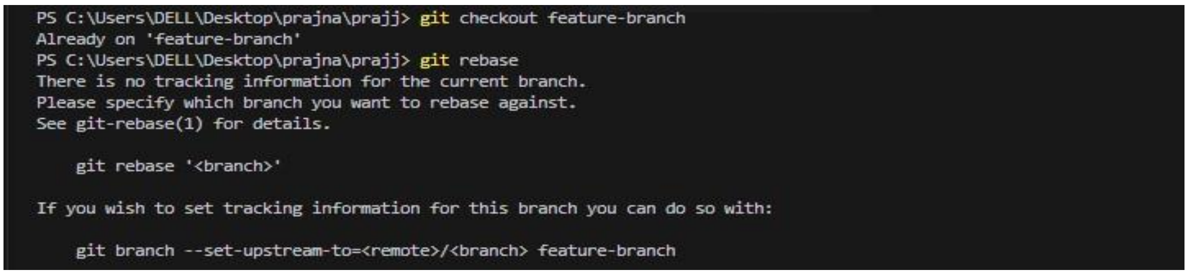


```
PS C:\Users\DELL\Desktop\prajna> git clone https://github.com/prajnar25/mod6.git
Cloning into 'mod6'...
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

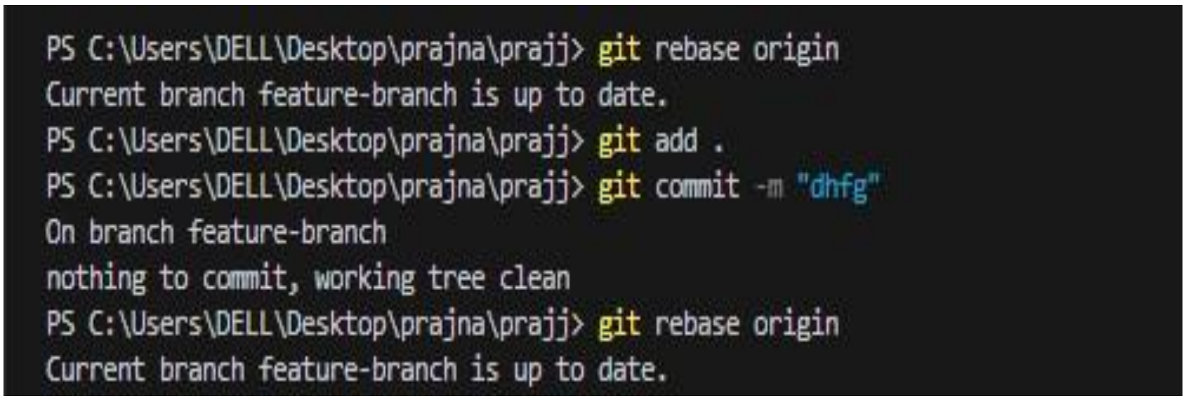
git checkout -b feature-branch

git fetch origin

git rebase origin/<branch_name>



```
PS C:\Users\DELL\Desktop\prajna\prajj> git checkout feature-branch
Already on 'feature-branch'
PS C:\Users\DELL\Desktop\prajna\prajj> git rebase
There is no tracking information for the current branch.
Please specify which branch you want to rebase against.
See git-rebase(1) for details.

    git rebase '<branch>'

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=<remote>/<branch> feature-branch
```



```
PS C:\Users\DELL\Desktop\prajna\prajj> git rebase origin
Current branch feature-branch is up to date.
PS C:\Users\DELL\Desktop\prajna\prajj> git add .
PS C:\Users\DELL\Desktop\prajna\prajj> git commit -m "dhfg"
On branch feature-branch
nothing to commit, working tree clean
PS C:\Users\DELL\Desktop\prajna\prajj> git rebase origin
Current branch feature-branch is up to date.
```

EXPERIMENT No.04: Git Tags and Releases

4.1 Objective

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

4.2 System Configuration

Windows 10, Linux, Mac

4.3 Pre-Requisite

Install Git bash and make required settings.

4.4 Introduction

git log

to view the commit history and find the commit ID you want to tag. Each commit is identified by a unique hash. git
tag v1.0 <commit_id>

git tag followed by the tag name (e.g., "v1.0") and the commit ID to create a lightweight tag. Lightweight tags are simply pointers to specific commits and do not contain additional metadata like annotated tags.

git show v1.0

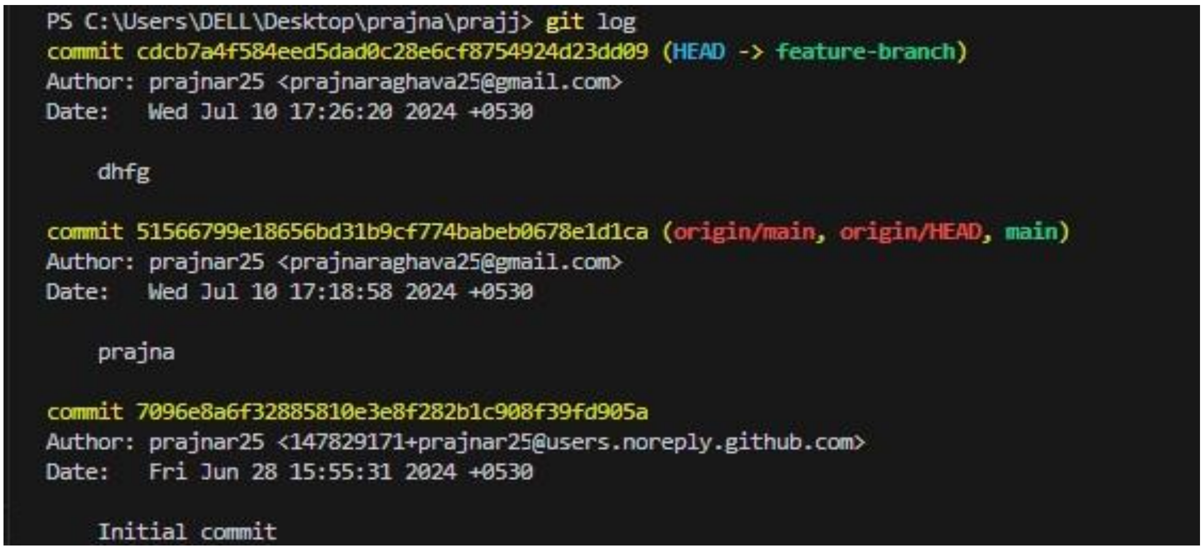
Use git show followed by the tag name to verify that the tag was created correctly and is pointing to the desired commit. This command will display the details of the tag, including the commit it points to.

git push origin v1.0

If you want to share the tag with others, you can push it to a remote repository using git push. This step is optional and depends on your workflow and the need to share the tag with others.

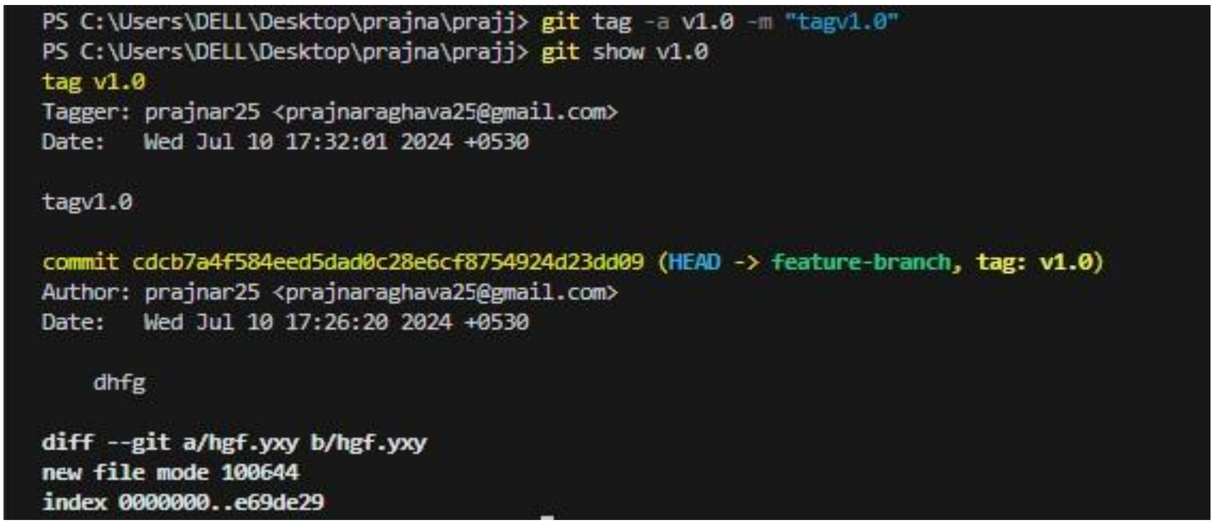
4.5 Procedure and Results

git log



```
PS C:\Users\DELL\Desktop\prajna\prajj> git log
commit cdcb7a4f584eed5dad0c28e6cf8754924d23dd09 (HEAD -> feature-branch)
Author: prajnar25 <prajnaraghava25@gmail.com>
Date:   Wed Jul 10 17:26:20 2024 +0530

    dhfg

commit 51566799e18656bd31b9cf774babeb0678e1d1ca (origin/main, origin/HEAD, main)
Author: prajnar25 <prajnaraghava25@gmail.com>
Date:   Wed Jul 10 17:18:58 2024 +0530

    prajna

commit 7096e8a6f32885810e3e8f282b1c908f39fd905a
Author: prajnar25 <147829171+prajnar25@users.noreply.github.com>
Date:   Fri Jun 28 15:55:31 2024 +0530

    Initial commit
```

git tag -a v1.0 -m “Tag v1.0” git show v1.0



```
PS C:\Users\DELL\Desktop\prajna\prajj> git tag -a v1.0 -m "tagv1.0"
PS C:\Users\DELL\Desktop\prajna\prajj> git show v1.0
tag v1.0
Tagger: prajnar25 <prajnaraghava25@gmail.com>
Date:   Wed Jul 10 17:32:01 2024 +0530

tagv1.0

commit cdcb7a4f584eed5dad0c28e6cf8754924d23dd09 (HEAD -> feature-branch, tag: v1.0)
Author: prajnar25 <prajnaraghava25@gmail.com>
Date:   Wed Jul 10 17:26:20 2024 +0530

    dhfg

diff --git a/hgf.yxy b/hgf.yxy
new file mode 100644
index 0000000..e69de29
```

EXPERIMENT No.05: Advanced Git Operations

5.1 Objective

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

5.2 System Configuration

Windows 10, Linux, Mac

5.3 Pre-Requisite

Install Git bash and make required settings.

5.4 Introduction

git checkout -b source-branch:

Creates and switches to a new branch named "source-branch."
Makes some change filename: Edits a file in the working directory.

git add . :

Stages all changes in the working directory for the next commit. git

push:

Pushes committed changes to the remote repository. git

commit -m "commit123":

Commits the staged changes with the given commit message. git

status:

Displays the status of changes as untracked, modified, or staged.
git log --oneline:

Shows a concise log of commits with their hash and short descriptions.

Copy hash of the commit:

Manually record the commit hash for reference. git
checkout -m main:

Switches to the "main" branch, assuming it already exists.

git cherry-pick cb0249:

Applies the changes from a specific commit (cb0249) to the current branch. git cherry-pick 5ei3ei...2b:

Picks a range of commits and applies them to the current branch.

git cherry-pick --continue:

Continues the cherry-pick process after resolving conflicts.

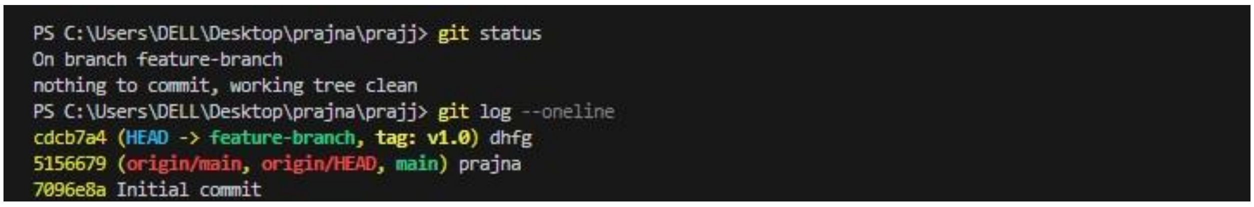
git log --oneline:

Displays an updated log after cherry-picking, showing the new commit(s).

5.5 Procedure and Results

git status

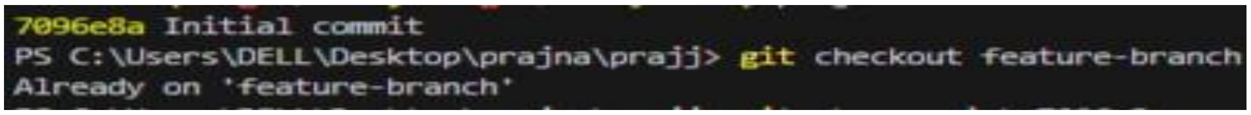
git log –oneline



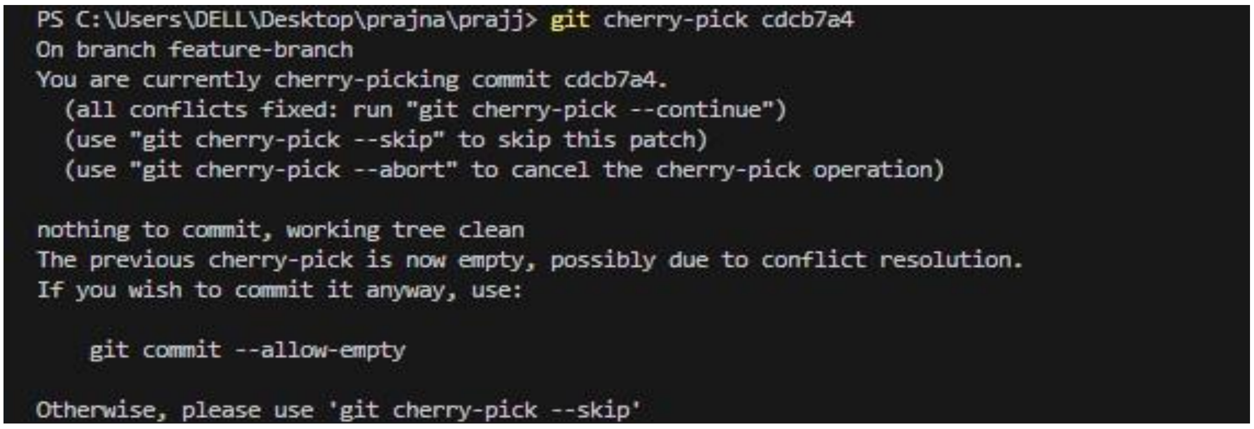
```
PS C:\Users\DELL\Desktop\prajna\prajj> git status
On branch feature-branch
nothing to commit, working tree clean
PS C:\Users\DELL\Desktop\prajna\prajj> git log --oneline
cdcb7a4 (HEAD -> feature-branch, tag: v1.0) dhfg
5156679 (origin/main, origin/HEAD, main) prajna
7096e8a Initial commit
```

git checkout feature-branch

git cherry-pick <hash>



```
7096e8a Initial commit
PS C:\Users\DELL\Desktop\prajna\prajj> git checkout feature-branch
Already on 'feature-branch'
```



```
PS C:\Users\DELL\Desktop\prajna\prajj> git cherry-pick cdcb7a4
On branch feature-branch
You are currently cherry-picking commit cdcb7a4.
  (all conflicts fixed: run "git cherry-pick --continue")
  (use "git cherry-pick --skip" to skip this patch)
  (use "git cherry-pick --abort" to cancel the cherry-pick operation)

nothing to commit, working tree clean
The previous cherry-pick is now empty, possibly due to conflict resolution.
If you wish to commit it anyway, use:

    git commit --allow-empty

Otherwise, please use 'git cherry-pick --skip'
```

EXPERIMENT No.06: Analzing and Changing Git History

6.1 Objective

a) Given a commit ID, how would you use Git to view the details of that specific commit, including the author, date, and commit message?
b) Write the command to list all commits made by the author "JohnDoe" between "2023-01-01" and "2023-12-31."
c) Write the command to display the last five commits in the repository's history.
d) Write the command to undo the changes introduced by the commit with the ID "abc123".

6.2 Software Required

Windows 10, Linux, Mac

6.3 Pre-Requisite

Install Git bash and make required settings.

6.4 Introduction

```
git show <commit_id>
```

This command displays the details of a specific commit, including the commit message, author, date, and the changes made in the commit. It's useful for reviewing the details of a particular commit in your Git history.

```
git log --author=JohnDoe --after=2023-01-01 --before=2023-12-31
```

This command lists all commits made by the author "JohnDoe" between the dates "2023-01-01" and "2023-12-31". It's helpful for filtering the commit history based on authorship and date ranges.

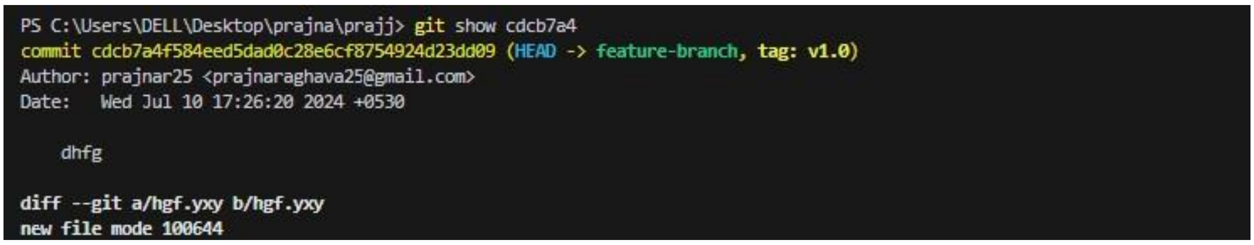
```
git log -n 5
```

This command displays the last five commits in the repository's history. It's useful for quickly viewing recent changes and understanding the recent development activity in the repository. git revert abc123

This command creates a new commit that undoes the changes introduced by the commit with the ID "abc123". It's a safe way to undo changes without altering the commit history, as it creates a new commit that reflects the changes being reverted.

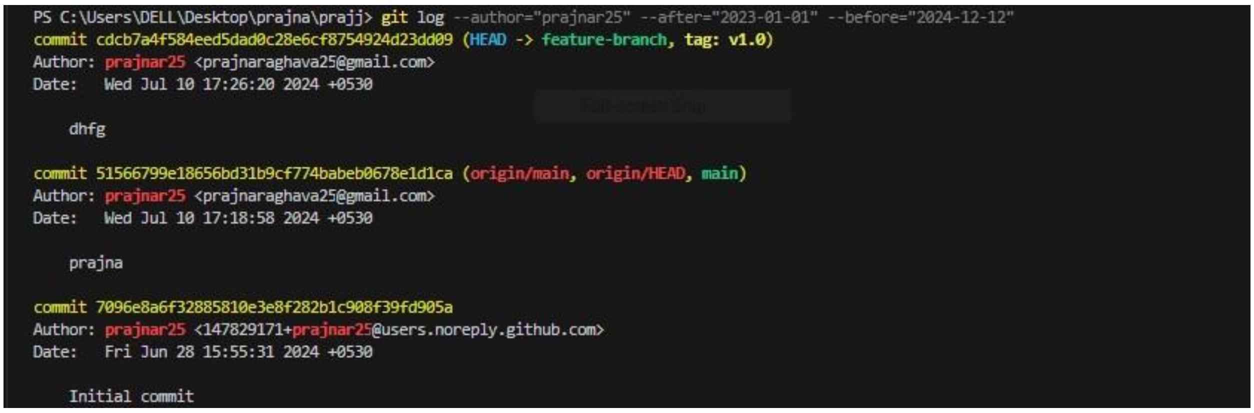
6.5 Procedure and Results

git show <commit_id>



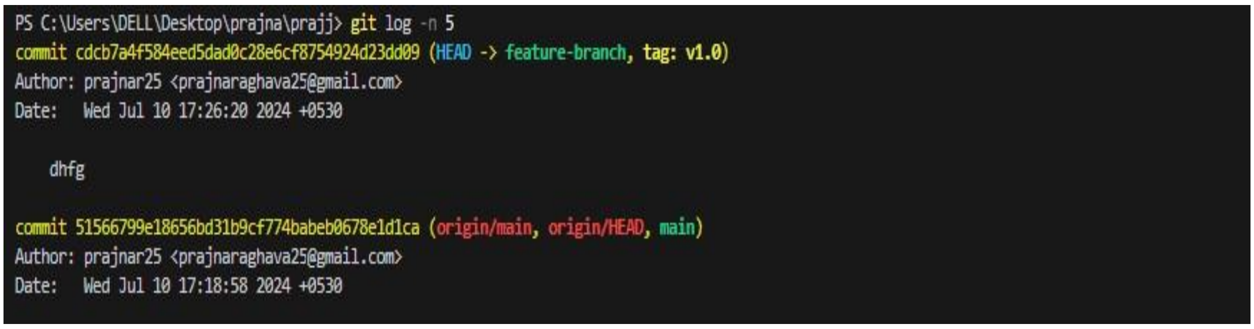
```
PS C:\Users\DELL\Desktop\prajna\prajj> git show cdcb7a4
commit cdcb7a4f584eed5dad0c28e6cf8754924d23dd09 (HEAD -> feature-branch, tag: v1.0)
Author: prajnar25 <prajnaraghava25@gmail.com>
Date:   Wed Jul 10 17:26:20 2024 +0530

    dhfg

diff --git a/hgf.yxy b/hgf.yxy
new file mode 100644
```

git log --author=”JohnDoe” --after=”2023-01-01” --before=”2023-12-31”



```
PS C:\Users\DELL\Desktop\prajna\prajj> git log --author="prajnar25" --after="2023-01-01" --before="2024-12-12"
commit cdcb7a4f584eed5dad0c28e6cf8754924d23dd09 (HEAD -> feature-branch, tag: v1.0)
Author: prajnar25 <prajnaraghava25@gmail.com>
Date:   Wed Jul 10 17:26:20 2024 +0530

    dhfg

commit 51566799e18656bd31b9cf774babeb0678e1d1ca (origin/main, origin/HEAD, main)
Author: prajnar25 <prajnaraghava25@gmail.com>
Date:   Wed Jul 10 17:18:58 2024 +0530

    prajna

commit 7096e8a6f32885810e3e8f282b1c908f39fd905a
Author: prajnar25 <147829171+prajnar25@users.noreply.github.com>
Date:   Fri Jun 28 15:55:31 2024 +0530

    Initial commit
```

git log -n 5



```
PS C:\Users\DELL\Desktop\prajna\prajj> git log -n 5
commit cdcb7a4f584eed5dad0c28e6cf8754924d23dd09 (HEAD -> feature-branch, tag: v1.0)
Author: prajnar25 <prajnaraghava25@gmail.com>
Date:   Wed Jul 10 17:26:20 2024 +0530

    dhfg

commit 51566799e18656bd31b9cf774babeb0678e1d1ca (origin/main, origin/HEAD, main)
Author: prajnar25 <prajnaraghava25@gmail.com>
Date:   Wed Jul 10 17:18:58 2024 +0530
```