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

SAMEEKSHA S
4GW24CI035

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

(BCS358C)

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 thecurrent.

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".

Git Commands List

Git is a popular version control system used for tracking change in software development projects. Here's a list of common Git commands along with brief explanations:

1. git init: Initializes a new Git repository in the current directory.

2. git clone <repository URL>: Creates a copy of a remote repository on your local machine.

3. git add <file>: Stages a file to be committed, marking it for tracking in the next commit.

4. git commit -m "message": Records the changes you've staged with a descriptive commit message.

5. git status: Shows the status of your working directory and the files that have been modified or staged.

6. git log: Displays a log of all previous commits, including commit hashes, authors, dates, and commit messages.

7. git diff: Shows the differences between the working directory and the last committed version.

8. git branch: Lists all branches in the repository and highlights the currently checked-out branch.

9. git branch <branchname>: Creates a new branch with the specified name.

10. git checkout <branchname>: Switches to a different branch.

11. git merge <branchname>: Merges changes from the specified branch into the currently checked-out branch.

12. git pull: Fetches changes from a remote repository and merges them into the current branch.

13. git push: Pushes your local commits to a remote repository.

14. git remote: Lists the remote repositories that your local repository is connected to.

15. git fetch: Retrieves changes from a remote repository without merging them.

16. git reset <file>: Unstages a file that was previously staged for commit.

17. git reset --hard <commit>: Resets the branch to a specific commit, discarding all changes after that commit.

18. git stash: Temporarily saves your changes to a "stash" so you can switch branches without committing or losing your work.

19. git tag: Lists and manages tags (usually used for marking specific points in history, like releases).

20. git blame <file>: Shows who made each change to a file and when.

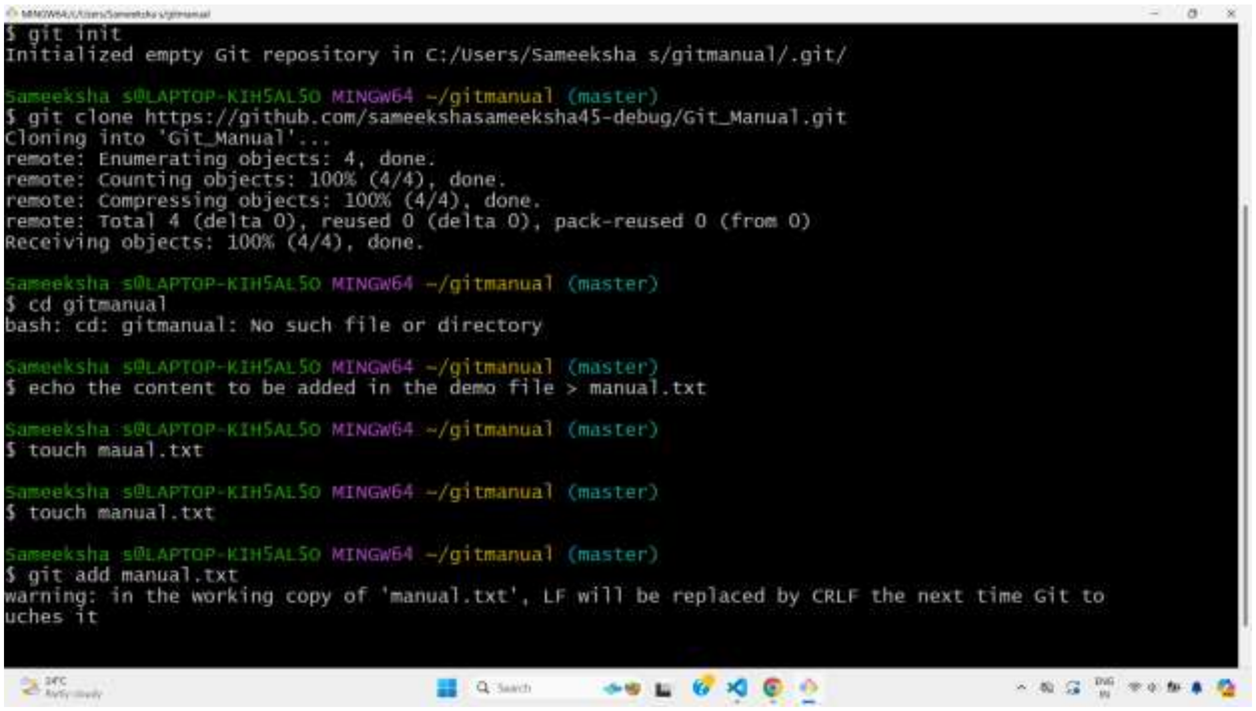
21. git rm <file>: Removes a file from both your working directory and the Git repository.

22. git mv <oldfile> <newfile>: Renames a file and stages the change.

These are some of the most common Git commands, but Git offers a wide range of features and options for more advanced usage. You can use git --help followed by the command name
to get more information about any specific command e.g., git help commit.

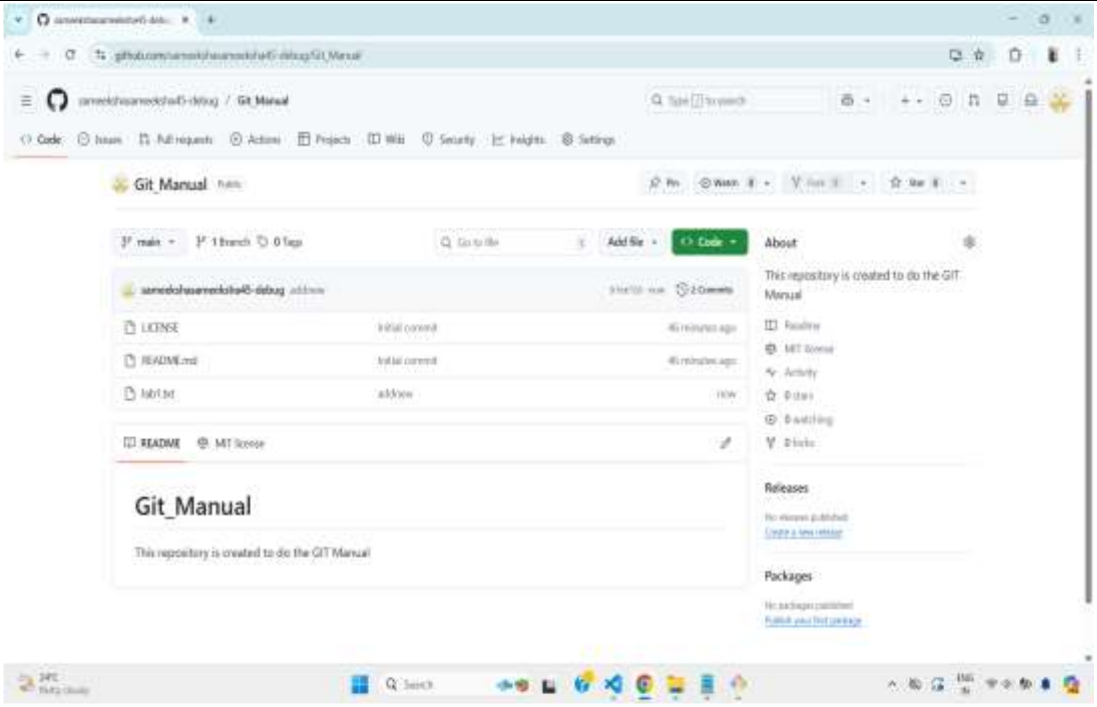
GIT REPO URL:-

https://github.com/sameekshasameeksha45-debug/projectlabos



```
$ git init
Initialized empty Git repository in C:/Users/Sameeksha s/gitmanual/.git/

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/gitmanual (master)
$ git clone https://github.com/sameekshasameeksha45-debug/Git_Manual.git
Cloning into 'Git_Manual'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (4/4), done.

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/gitmanual (master)
$ cd gitmanual
bash: cd: gitmanual: No such file or directory

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/gitmanual (master)
$ echo the content to be added in the demo file > manual.txt

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/gitmanual (master)
$ touch maual.txt

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/gitmanual (master)
$ touch manual.txt

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/gitmanual (master)
$ git add manual.txt
warning: in the working copy of 'manual.txt', LF will be replaced by CRLF the next time Git to
uches it
```

gitinit
notepad lab1.txt
git add lab1.txt
git status
git commit -m "Experiment 1: Initial repository setup and first commit"
git remote add origin https://github.com/sameekshasameeksha45-debug/projectlabos
git branch -M main
git pull --rebase origin main
git push -u origin main



sameekshasameeksha45-debug / Git_Manual

Git_Manual Public

main · 1 Branch · 0 Tags

sameekshasameeksha45-debug addnew · 2 Commits

LICENSE	Initial commit	45 minutes ago
README.md	Initial commit	45 minutes ago
lab1.txt	addnew	now

README · MIT license

Git_Manual

This repository is created to do the GIT Manual

About

This repository is created to do the GIT Manual

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

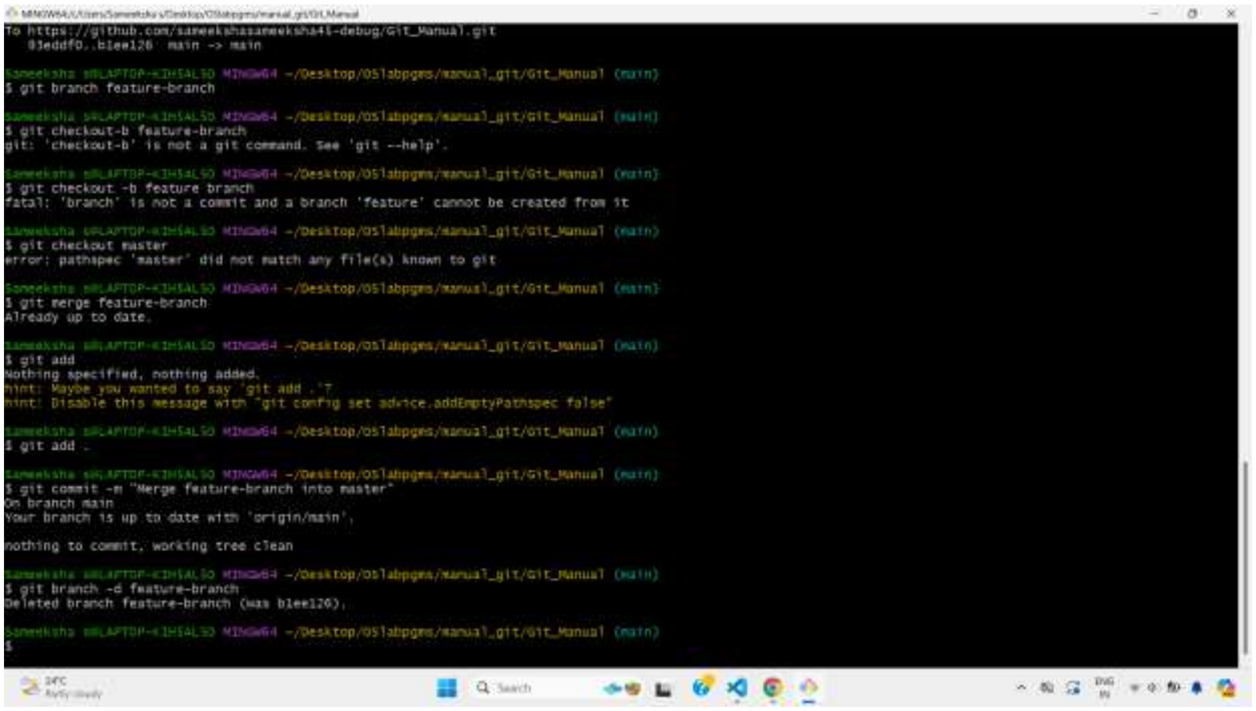
EXPERIMENT 2: CREATING AND MERGING BRANCHES

PROCEDURE

1. Open Git Bash and move to the project directory.
2. Switch to the main branch using the git checkout command.
3. Create a new branch named feature-branch and switch to it.
4. Modify an existing file using Notepad.
5. Add the modified file to the staging area.
6. Commit the changes in the feature branch.
7. Switch back to the main branch.
8. Merge the feature-branch into the main branch.

COMMANDS USED:

```
git checkout main
git checkout -b feature-branch
notepad lab1.txt
git add lab1.txt
git commit -m "Experiment 2: Changes in feature branch"
git checkout main
git merge feature-branch
```



```
To https://github.com/sameekshasameeksha45-debug/Git_Manual.git
   03eddf0..b1ee126  main -> main

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual (main)
$ git branch feature-branch

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual (main)
$ git checkout-b feature-branch
git: 'checkout-b' is not a git command. See 'git --help'.

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual (main)
$ git checkout -b feature branch
fatal: 'branch' is not a commit and a branch 'feature' cannot be created from it

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual (main)
$ git checkout master
error: pathspec 'master' did not match any file(s) known to git

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual (main)
$ git merge feature-branch
Already up to date.

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual (main)
$ git add
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'?
hint: Disable this message with "git config set advice.addEmptyPathspec false"

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual (main)
$ git add .

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual (main)
$ git commit -m "Merge feature-branch into master"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual (main)
$ git branch -d feature-branch
Deleted branch feature-branch (was b1ee126).

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual (main)
$
```

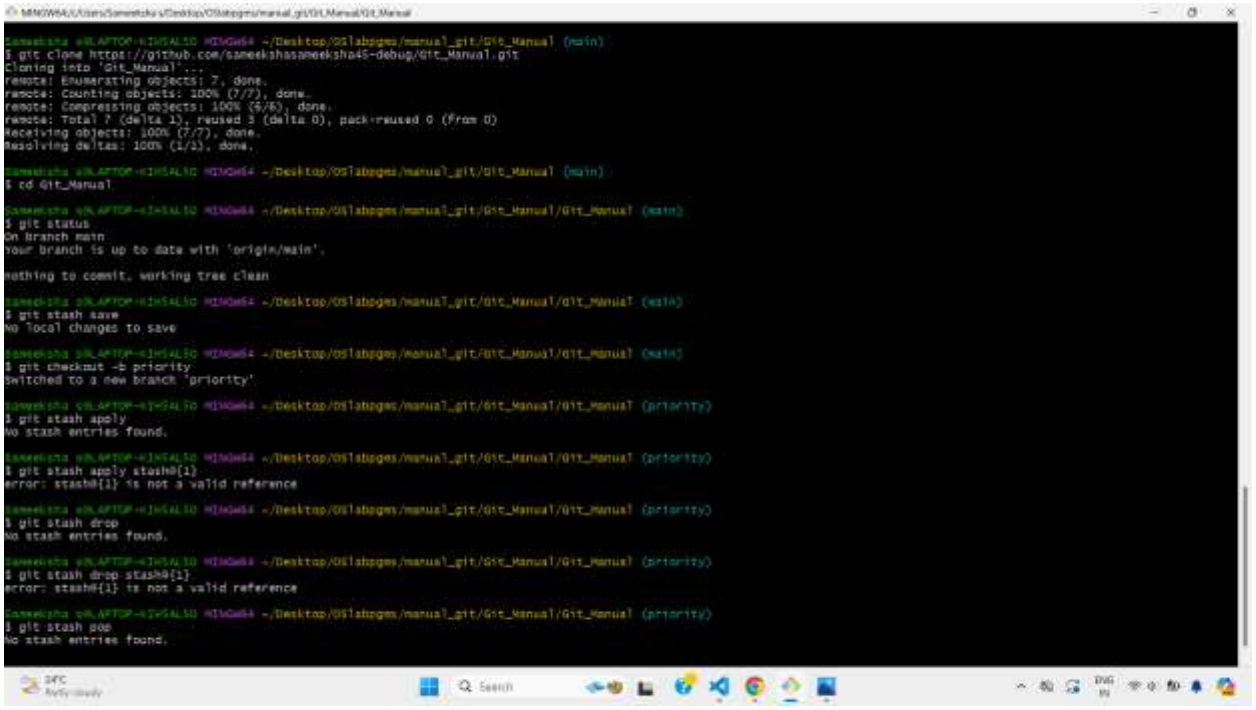
EXPERIMENT 3: CREATING AND MERGING BRANCHES

PROCEDURE:

1. Open Git Bash and move to the project directory.
2. Modify an existing file using Notepad.
3. Check the repository status using the git status command.
4. Save the uncommitted changes using the git stash command.
5. Verify that the working directory is clean.
6. Switch to another branch.
7. Apply the stashed changes using the git stash apply command.
8. Check the repository status again.

COMMANDS USED:

notepad lab1.txt , git status , git stash , git checkout feature1 , git stash apply , git status

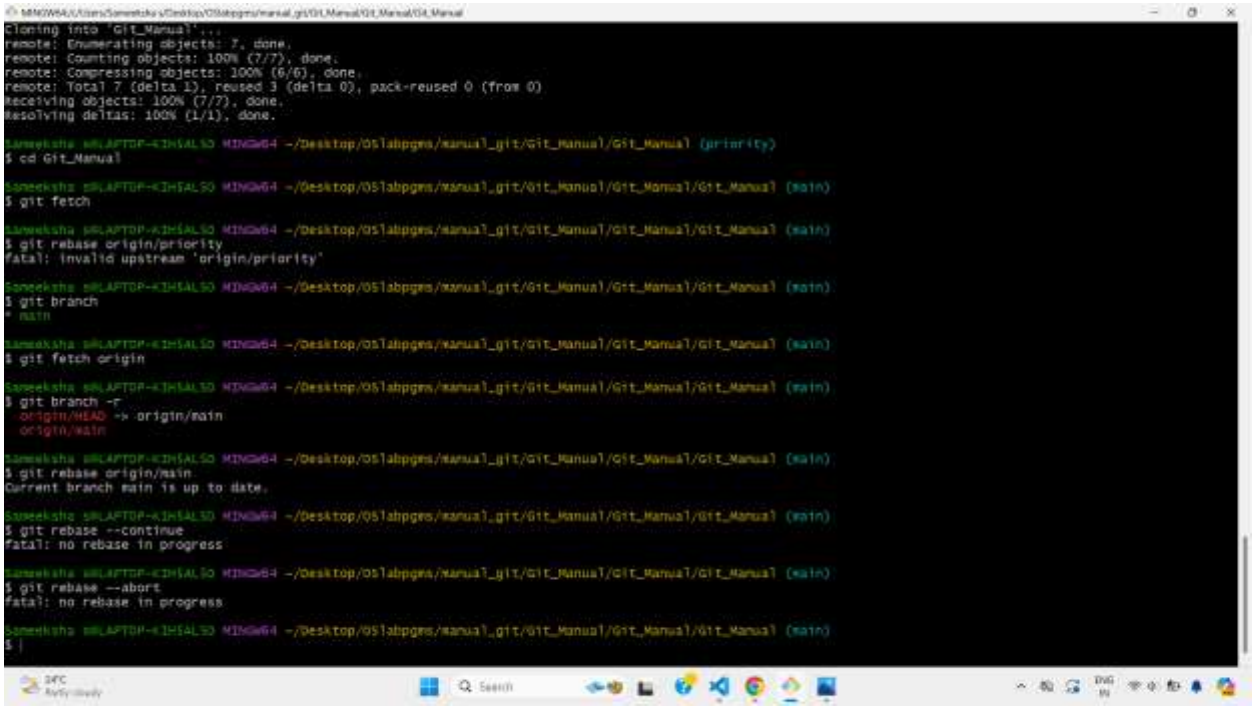


```
Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual (main)
$ git clone https://github.com/sameekshasameeksha45-debug/Git_Manual.git
Cloning into 'Git_Manual'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 1), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (7/7), done.
Resolving deltas: 100% (1/1), done.

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual (main)
$ cd Git_Manual

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual (main)
$ git stash save
No local changes to save

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual (main)
$ git checkout -b priority
Switched to a new branch 'priority'

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual (priority)
$ git stash apply
No stash entries found.

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual (priority)
$ git stash apply stash@{1}
error: stash@{1} is not a valid reference

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual (priority)
$ git stash drop
No stash entries found.

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual (priority)
$ git stash drop stash@{1}
error: stash@{1} is not a valid reference

Sameeksha @LAPTOP-KIHSAL5D MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual (priority)
$ git stash pop
No stash entries found.
```

EXPERIMENT 4: COLLABORATIONS AND REPORT REPOSITORIES

PROCEDURE:

1. The terminal was navigated to the Desktop using the cd command.
2. The remote GitHub repository was copied to the local system using the git clone command.
3. The cloned repository folder was accessed using the cd command.
4. The files present in the repository were displayed using the ls command.
5. The current status of the repository was checked using the git status command.



```
Cloning into 'Git_Manual'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 1), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (7/7), done.
Resolving deltas: 100% (1/1), done.

Sameeksha @LAPTOP-KIHSALSD MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual (priority)
$ cd Git_Manual

Sameeksha @LAPTOP-KIHSALSD MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git fetch

Sameeksha @LAPTOP-KIHSALSD MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git rebase origin/priority
fatal: invalid upstream 'origin/priority'

Sameeksha @LAPTOP-KIHSALSD MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git branch
* main

Sameeksha @LAPTOP-KIHSALSD MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git fetch origin

Sameeksha @LAPTOP-KIHSALSD MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git branch -r
  origin/HEAD -> origin/main
  origin/main

Sameeksha @LAPTOP-KIHSALSD MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git rebase origin/main
Current branch main is up to date.

Sameeksha @LAPTOP-KIHSALSD MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git rebase --continue
fatal: no rebase in progress

Sameeksha @LAPTOP-KIHSALSD MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git rebase --abort
fatal: no rebase in progress

Sameeksha @LAPTOP-KIHSALSD MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ |
```

EXPERIMENT 5: COLLABORATIONS AND REPORT REPOSITORIES

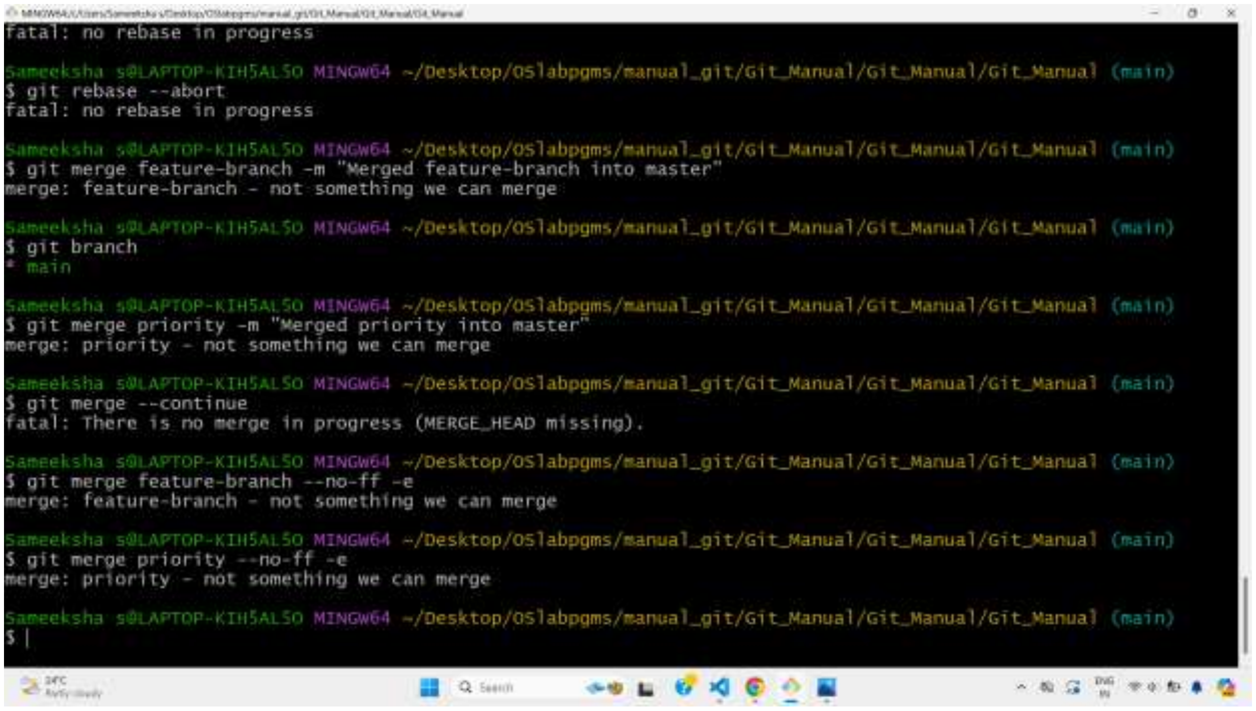
PROCEDURE:

1. The terminal was navigated to the local GitHub repository directory.
2. The current branch and working tree status were verified using the git status command.
3. The latest changes from the remote repository were fetched using the git fetch origin command.
4. The local branch was synchronized with the remote main branch using the git rebase origin/main command.
5. The repository status was checked again to ensure that the working tree was clean and up to date.

COMMANDS USED:

cd ~

/Desktop/github , git status , git status , git fetch origin, git rebase origin/main



```
fatal: no rebase in progress

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git rebase --abort
fatal: no rebase in progress

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git merge feature-branch -m "Merged feature-branch into master"
merge: feature-branch - not something we can merge

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git branch
* main

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git merge priority -m "Merged priority into master"
merge: priority - not something we can merge

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git merge --continue
fatal: There is no merge in progress (MERGE_HEAD missing).

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git merge feature-branch --no-ff -e
merge: feature-branch - not something we can merge

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git merge priority --no-ff -e
merge: priority - not something we can merge

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ |
```

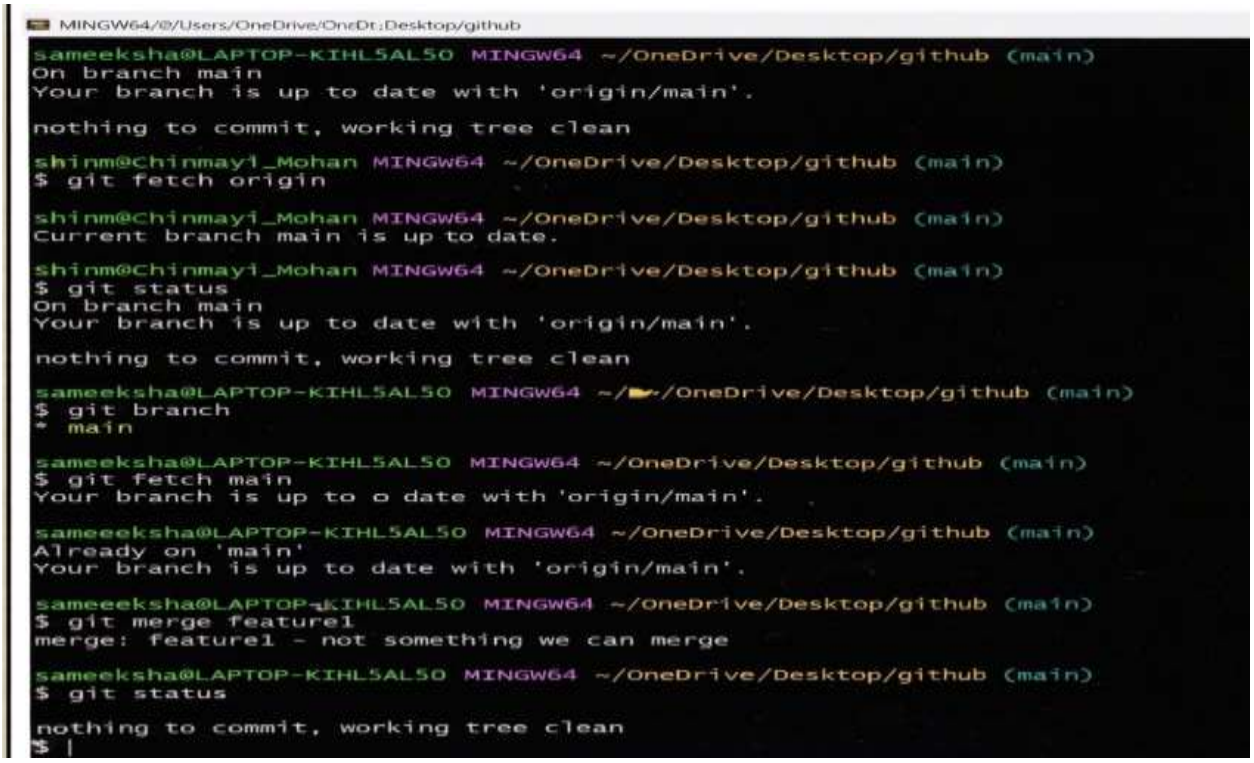
EXPERIMENT 6: COLLABORATION AND REMOTE REPOSITORIES

PROCEDURE:

1. The list of available branches was displayed using the git branch command.
2. The current working branch was switched to the main branch using git checkout main.
3. The changes from the feature branch were merged into the main branch using the git merge feature1 command.
4. The status of the repository was verified using the git status command to ensure successful merge.

COMMANDS USED:

git branch, git checkout main, git , git merge feature1, git status



MINGW64/@/Users/OneDrive/OneDt;Desktop/github

```
sameeksha@LAPTOP-KIHL5AL50 MINGW64 ~/OneDrive/Desktop/github (main)
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

shinm@Chinmayi_Mohan MINGW64 ~/OneDrive/Desktop/github (main)
$ git fetch origin

shinm@Chinmayi_Mohan MINGW64 ~/OneDrive/Desktop/github (main)
Current branch main is up to date.

shinm@Chinmayi_Mohan MINGW64 ~/OneDrive/Desktop/github (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

sameeksha@LAPTOP-KIHL5AL50 MINGW64 ~/OneDrive/Desktop/github (main)
$ git branch
* main

sameeksha@LAPTOP-KIHL5AL50 MINGW64 ~/OneDrive/Desktop/github (main)
$ git fetch main
Your branch is up to o date with 'origin/main'.

sameeeksha@LAPTOP-KIHL5AL50 MINGW64 ~/OneDrive/Desktop/github (main)
Already on 'main'
Your branch is up to date with 'origin/main'.

sameeeksha@LAPTOP-KIHL5AL50 MINGW64 ~/OneDrive/Desktop/github (main)
$ git merge feature1
merge: feature1 - not something we can merge

sameeksha@LAPTOP-KIHL5AL50 MINGW64 ~/OneDrive/Desktop/github (main)
$ git status

nothing to commit, working tree clean
$ |
```

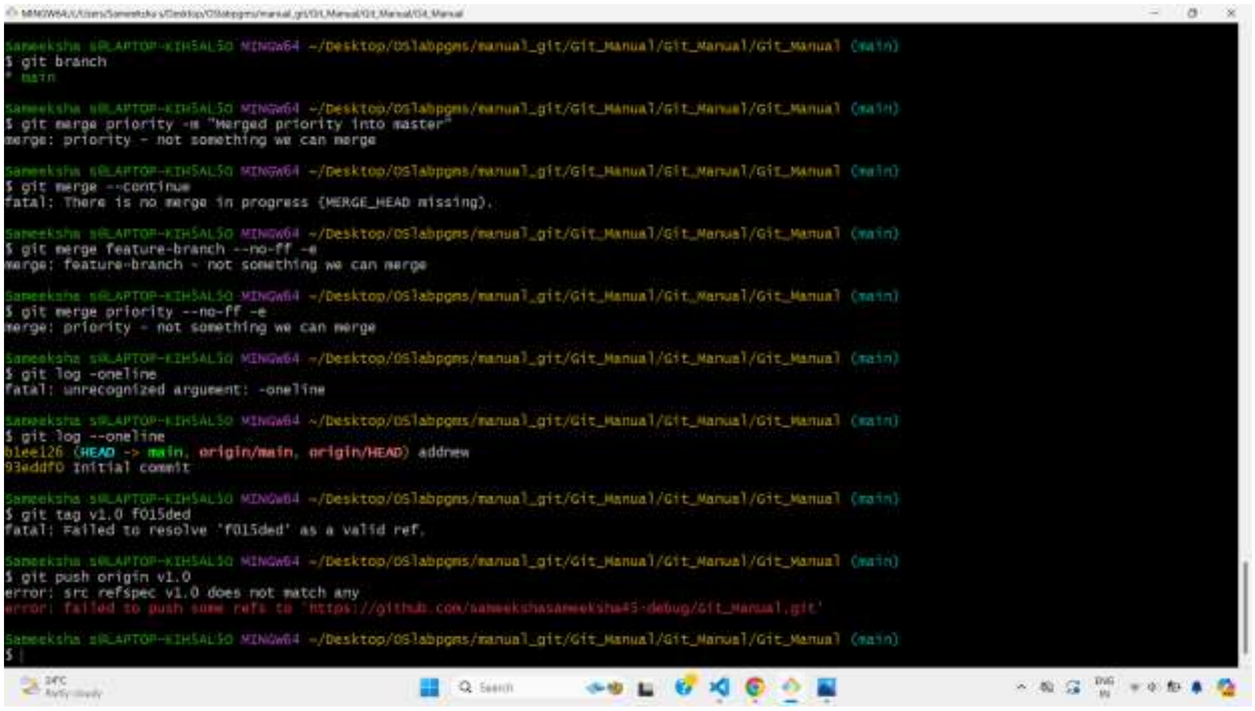
EXPERIMENT 7: GIT TAGS AND RELEASES

PROCEDURE:

1. The working branch was verified and switched to the main branch using the git checkout main command.
2. The current repository status was checked using git status.
3. A tag named v1.0 was created using the git tag command to mark an important version of the project.
4. The list of available tags was displayed using the git tag command.
5. The commit history was viewed using git log --oneline to confirm that the tag was attached to the correct commit.

COMMANDS USED:

git c heckout main , git status, git tag v1.0, git tag , git log –oneline



```
Sameeksha S@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git branch
* main

Sameeksha S@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git merge priority -m "Merged priority into master"
merge: priority - not something we can merge

Sameeksha S@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git merge --continue
fatal: There is no merge in progress (MERGE_HEAD missing).

Sameeksha S@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git merge feature-branch --no-ff -e
merge: feature-branch - not something we can merge

Sameeksha S@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git merge priority --no-ff -e
merge: priority - not something we can merge

Sameeksha S@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git log -oneline
fatal: unrecognized argument: -oneline

Sameeksha S@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git log --oneline
b1ee126 (HEAD -> main, origin/main, origin/HEAD) addnew
93eddf0 Initial commit

Sameeksha S@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git tag v1.0 f015ded
fatal: Failed to resolve 'f015ded' as a valid ref.

Sameeksha S@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ git push origin v1.0
error: src refspec v1.0 does not match any
error: failed to push some refs to 'https://github.com/sameekshasameeksha45-debug/Git_Manual.git'

Sameeksha S@LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgms/manual_git/Git_Manual/Git_Manual/Git_Manual (main)
$ |
```

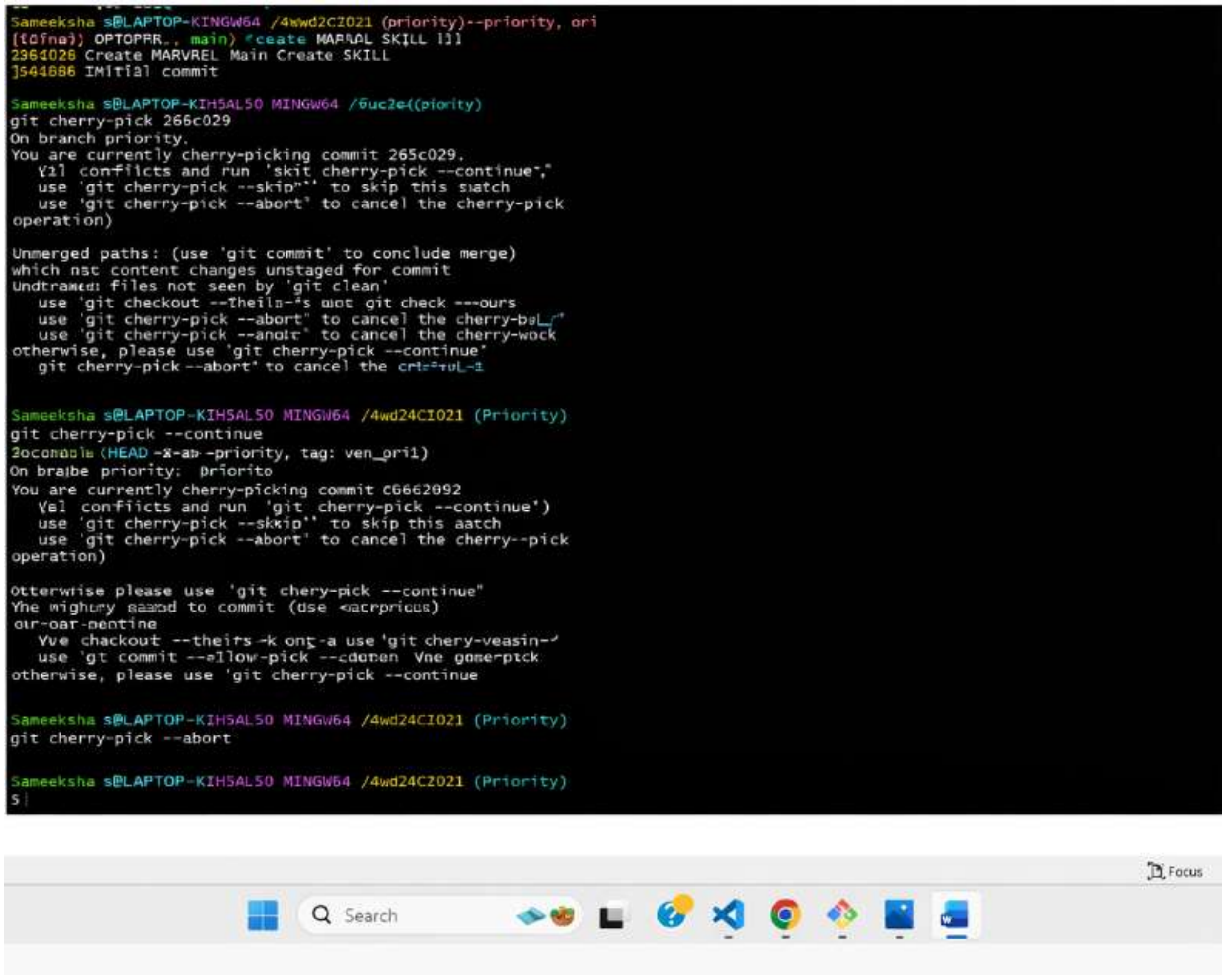
EXPERIMENT 8: ADVANCED GIT OPERATIONS

PROCEDURE:

1. The commit history was viewed using the git log --oneline command.
2. A new branch named feature1 was created using git checkout -b feature1.
3. The file lab1.txt was modified using Notepad.
4. The modified file was staged using the git add command.
5. The changes were committed using the git commit command.
6. The working branch was switched back to main using git checkout main.
7. The commit from the feature branch was applied to the main branch using the git cherry-pick command.
8. The commit history was verified to confirm successful cherry-pick.

COMMANDS USED:

git log –oneline ,
git checkout -b feature1
notepad lab1.txt
git add lab1.txt
git commit -m "Commit for cherry-pick experiment"
git checkout main
git cherry-pick feature1
git log –oneline



```
Sameeksha s@LAPTOP-KINGW64 /4wwd2CI021 (priority)--priority, ori
(tdfne)) OPTOPRR., main) *ceate MARNAL SKILL I1]
2364028 Create MARVREL Main Create SKILL
]544686 IMitial commit

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 /6uc2e((piority)
git cherry-pick 266c029
On branch priority.
You are currently cherry-picking commit 265c029.
   Y1l conflicts and run 'skit cherry-pick --continue","
   use 'git cherry-pick --skip"' to skip this satch
   use 'git cherry-pick --abort' to cancel the cherry-pick
operation)

Unmerged paths: (use 'git commit' to conclude merge)
which nst content changes unstaged for commit
Undtramed: files not seen by 'git clean'
   use 'git checkout --Theiln-'s mot git check ---ours
   use 'git cherry-pick --abort" to cancel the cherry-bsL"
   use 'git cherry-pick --anoit' to cancel the cherry-wock
otherwise, please use 'git cherry-pick --continue'
   git cherry-pick --abort' to cancel the [illegible]


Sameeksha s@LAPTOP-KIH5AL50 MINGW64 /4wd24CI021 (Priority)
git cherry-pick --continue
3ocomoole (HEAD -X-ab -priority, tag: ven_ori1)
On braibe priority:  Priorito
You are currently cherry-picking commit C6662092
   Vel confiicts and run 'git cherry-pick --continue')
   use 'git cherry-pick --skip'' to skip this aatch
   use 'git cherry-pick --abort' to cancel the cherry--pick
operation)

Otterwise please use 'git chery-pick --continue"
Yhe mighury saamd to commit (dse <acrpricus)
oir-oar-oentine
   Vve chackout --theirs -k ong-a use 'git chery-veasin-'
   use 'gt commit --allow-pick --cdoten Vne gomerptck
otherwise, please use 'git cherry-pick --continue


Sameeksha s@LAPTOP-KIH5AL50 MINGW64 /4wd24CI021 (Priority)
git cherry-pick --abort

Sameeksha s@LAPTOP-KIH5AL50 MINGW64 /4wd24C2021 (Priority)
5 |
```

EXPERIMENT 9: ANALYSING AND CHANGING GIT HISTORY

PROCEDURE:

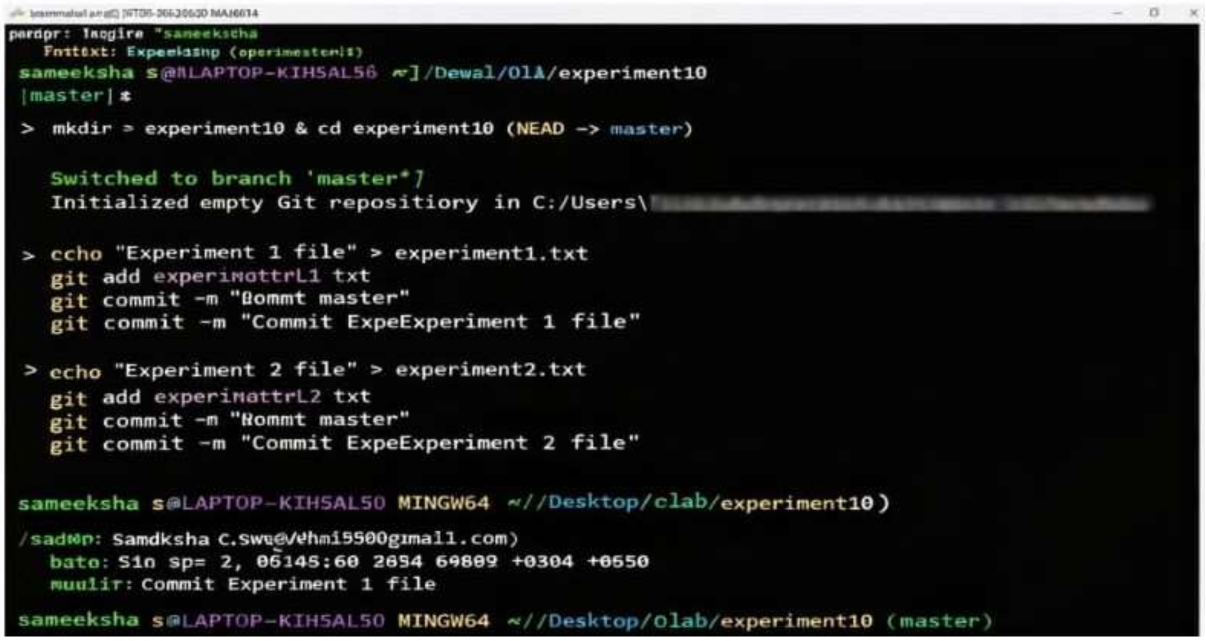
1. The commit history of the repository was viewed using the git log --oneline command to display concise commit information.
2. The complete commit history with author name, date, and commit message was viewed using the git log command.
3. A specific commit was selected from the history and its detailed information was displayed using the git show command.
4. The changes made in the selected commit were analyzed using the diff output.

COMMANDS USED:

git log –oneline ,

git log ,

git show <commit-id>

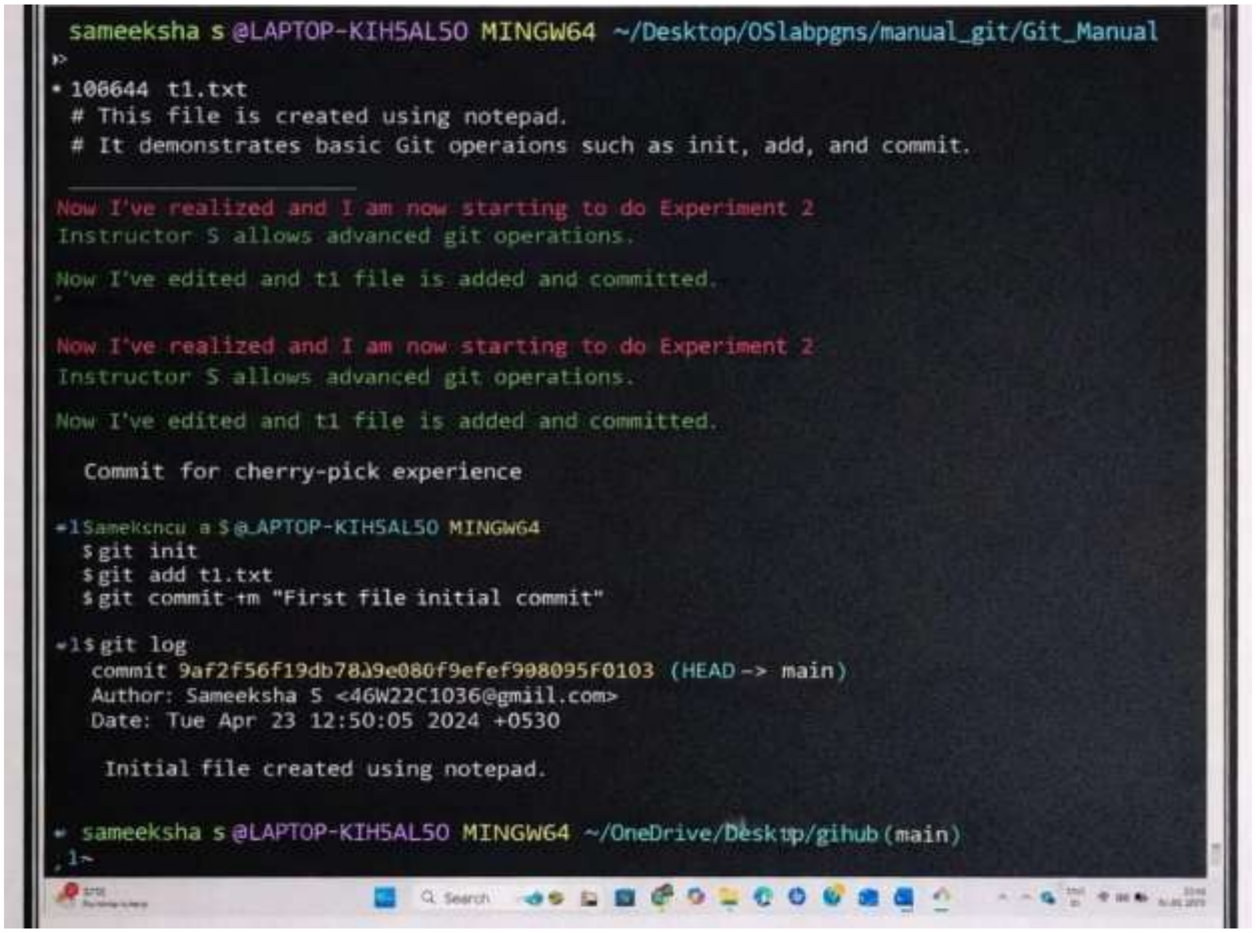


```
perdpr: Inogire "saneekscha
   Fnttext: Expeekisnp (operimesten!1)
sameeksha s@ℕLAPTOP-KIHSAL56 ~]/Dewal/OlÅ/experiment10
|master|$
>  mkdir = experiment10 & cd experiment10 (NEAD -> master)

   Switched to branch 'master*]
   Initialized empty Git repository in C:/Users\[redacted]

>  ccho "Experiment 1 file" > experiment1.txt
   git add experiмottrL1 txt
   git commit -m "ßommt master"
   git commit -m "Commit ExpeExperiment 1 file"

>  echo "Experiment 2 file" > experiment2.txt
   git add experiмattrL2 txt
   git commit -m "Rommt master"
   git commit -m "Commit ExpeExperiment 2 file"

sameeksha s@LAPTOP-KIHSAL50 MINGW64 ~//Desktop/clab/experiment10)
/sadмp: Samdksha C.Swų@Vehm15500gımall.com)
   bato: S1n sp= 2, 06145:60 2654 69809 +0304 +0650
   muulir: Commit Experiment 1 file
sameeksha s@LAPTOP-KIHSAL50 MINGW64 ~//Desktop/Olab/experiment10 (master)
```

EXPERIMENT 9: ANALYSING AND CHANGING GIT HISTORY

PROCEDURE:

1. The commit history of the repository was viewed using the git log --oneline command to display concise commit information.
2. The complete commit history with author name, date, and commit message was viewed using the git log command.
3. A specific commit was selected from the history and its detailed information was displayed using the git show command.
4. The changes made in the selected commit were analyzed using the diff output.



```
 sameeksha s @LAPTOP-KIH5AL50 MINGW64 ~/Desktop/OSlabpgns/manual_git/Git_Manual
»
• 106644  t1.txt
  # This file is created using notepad.
  # It demonstrates basic Git operaions such as init, add, and commit.

  ______________________
Now I've realized and I am now starting to do Experiment 2
Instructor S allows advanced git operations.

Now I've edited and t1 file is added and committed.

Now I've realized and I am now starting to do Experiment 2
Instructor S allows advanced git operations.

Now I've edited and t1 file is added and committed.

   Commit for cherry-pick experience

•1Samekshcu a $@LAPTOP-KIH5AL50 MINGW64
   $git init
   $git add t1.txt
   $git commit +m "First file initial commit"

•1$git log
    commit 9af2f56f19db78a9e080f9efef908095F0103 (HEAD -> main)
    Author: Sameeksha S <46W22C1036@gmiil.com>
    Date:  Tue Apr 23 12:50:05 2024 +0530

     Initial file created using notepad.

• sameeksha s @LAPTOP-KIH5AL50 MINGW64 ~/OneDrive/Desktop/gihub (main)
,1~
```

COMMANDS USED:
git log –oneline ,
git log ,
git show <commit-id>

EXPERIMENT 10: ANALYSING AND CHANGING GIT HISTORY

PROCEDURE:

1. The current status of the repository was checked using the git status command.
2. The file lab1.txt was modified using Notepad.
3. The repository status was checked again to confirm that the file was modified.
4. The modified file was restored to its last committed state using the git restore lab1.txt command.
5. The repository status was verified to ensure that the working tree was clean.

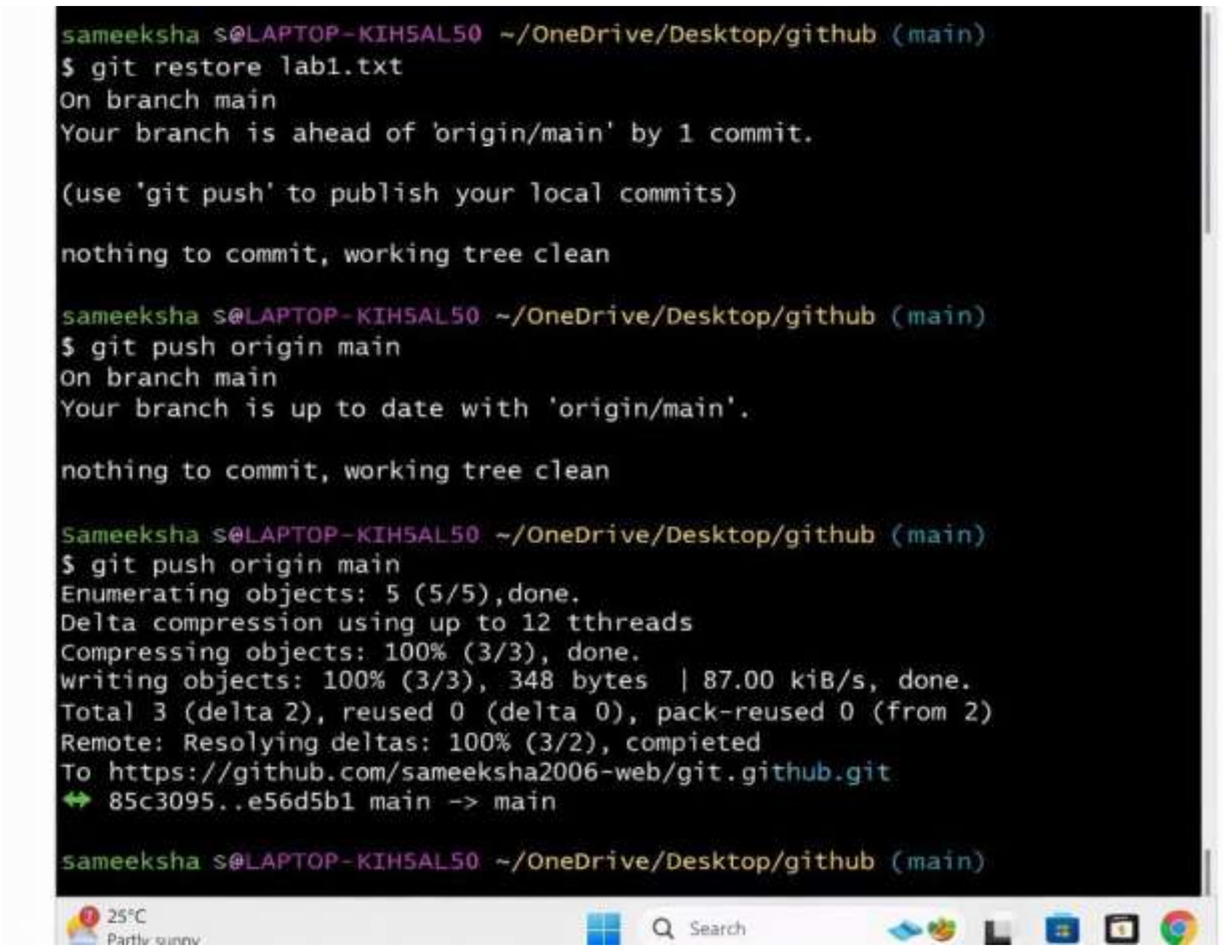
COMMANDS USED:

```
git status
notepad lab1.txt
git status
```

EXPERIMENT 11: ANALYSING AND CHANGING GIT HISTORY

PROCEDURE:

1. The current status of the local repository was checked using the git status command.
2. The committed changes were pushed to the remote GitHub repository using the git push origin main command.
3. After pushing, the status was verified to ensure the local and remote repositories were synchronized.
4. The updated files and commits were verified on the GitHub repository.



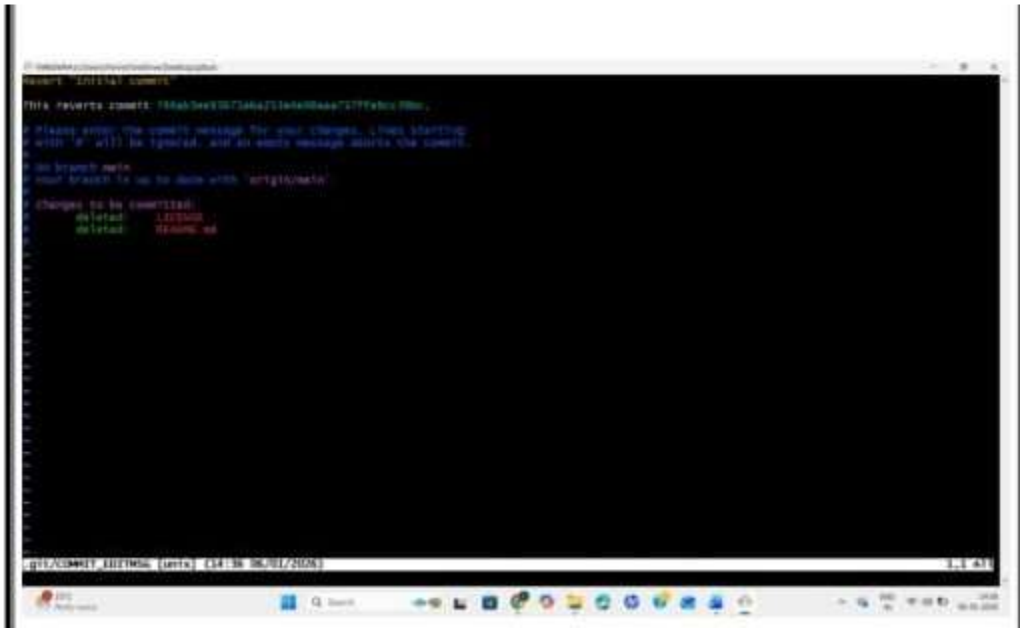
```
sameeksha s@LAPTOP-KIH5AL50 ~/OneDrive/Desktop/github (main)
$ git restore lab1.txt
On branch main
Your branch is ahead of 'origin/main' by 1 commit.

(use 'git push' to publish your local commits)

nothing to commit, working tree clean

sameeksha s@LAPTOP-KIH5AL50 ~/OneDrive/Desktop/github (main)
$ git push origin main
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Sameeksha s@LAPTOP-KIH5AL50 ~/OneDrive/Desktop/github (main)
$ git push origin main
Enumerating objects: 5 (5/5),done.
Delta compression using up to 12 tthreads
Compressing objects: 100% (3/3), done.
writing objects: 100% (3/3), 348 bytes  | 87.00 kiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 2)
Remote: Resolying deltas: 100% (3/2), compieted
To https://github.com/sameeksha2006-web/git.github.git
↔ 85c3095..e56d5b1 main -> main

sameeksha s@LAPTOP-KIH5AL50 ~/OneDrive/Desktop/github (main)
```

EXPERIMENT 12: ANALYSING AND CHANGING GIT HISTORY

PROCEDURE:

1. The commit history was viewed using the git log --oneline command.
2. The commit that needed to be undone was identified.
3. The git revert command was used to reverse the changes made by the selected commit.
4. A new revert commit was created without deleting previous commits.
5. The repository status and commit history were verified.
6. The revert commit was pushed to the remote GitHub repository

COMMANDS USED:



```
Revert "Initial commit"

This reverts commit [illegible].

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Your branch is up to date with 'origin/main'.
#
# Changes to be committed:
#       deleted:    LICENSE
#       deleted:    README.md
#
~
.git/COMMIT_EDITMSG [unix] (14:36 06/01/2026)                    1,1 All
```

git log –oneline, git revert <commit-id>, git status, git pus git push origin main