

Subject name: Source Code Management

Subject name: **Source code management**

Subject code: Subject code: CS181 **CS181**

Cluster: Cluster:

Gamma **Gamma**

Department: **DCSE**

Department: DCSE

**Submitted by:**

Kirpajeet Singh

2110990786

G17-B

**Submitted to:**

Dr. Priya Sadana



**Department of Computer Science & Engineering**

Chitkara University Institute of Engineering and Technology, Punjab

Jan- June (2022-23)



Institute/School Name	<b>Chitkara University Institute of Engineering and Technology</b>		
Department Name	<b>Department of Computer Science &amp; Engineering</b>		
Programme Name	<b>Bachelor of Engineering (B.E.), Computer Science &amp; Engineering</b>		
Course Name	<b>Source Code Management</b>	Session	<b>2022-23</b>
Course Code	<b>CS181</b>	Semester/Batch	<b>4th/2023</b>
Vertical Name	<b>Gamma</b>	Group No	<b>G17-B</b>
Course Coordinator	<b>Dr. Priya Sadana</b>		
Faculty Name	<b>Dr. Priya Sadana</b>		

## Submission

**Name:** Kirpajeet Singh

**Signature:** Kirpajeet Singh

**Date:** 10-05-23

i



## Table of Content

S. No.	Title	Page No.
1	Version control with Git	1-14
2	Final Result	15

3	Task 1.2	16-25
---	----------	-------

# 1. Version control with Git

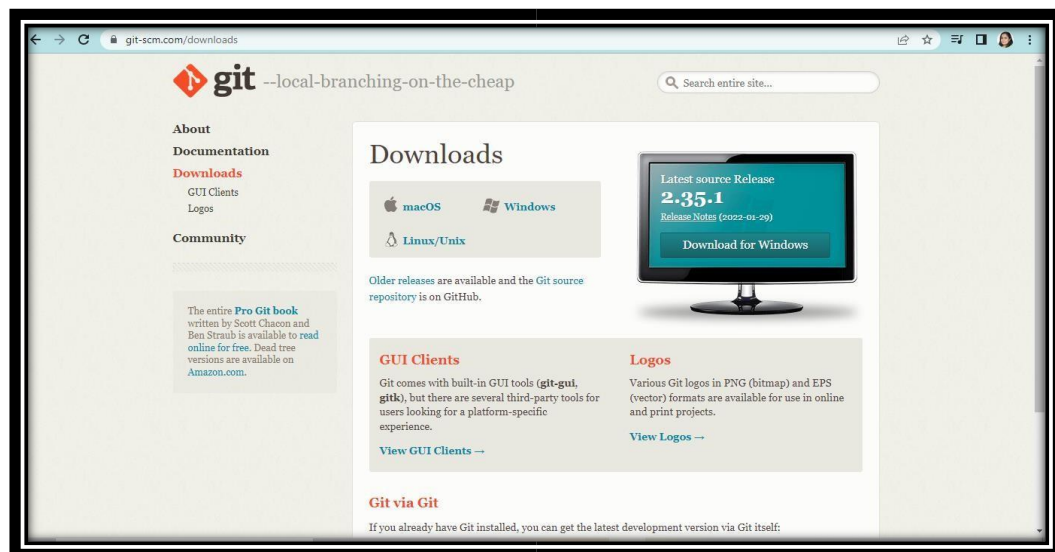
## Step 1 - Installing Git:

Before we start using Git, we have to make it available on your computer. Even if it's already Installed, it's probably a good idea to update to the latest version. We can either install it as a package or via another installer, or download the source code and compile it yourself.

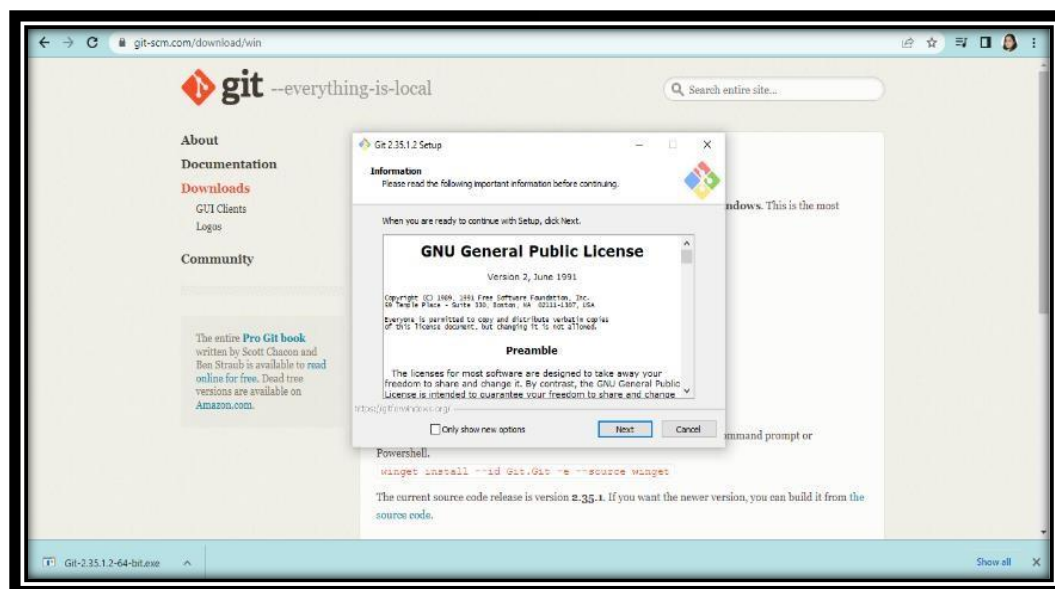
## Step 2 - Installing on windows:

There are also a few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to <https://git-scm.com/download/win> and the download will start automatically. Note that this is a project called Git for Windows, which is separate from Git itself; for more information on it, go to <https://gitforwindows.org>.

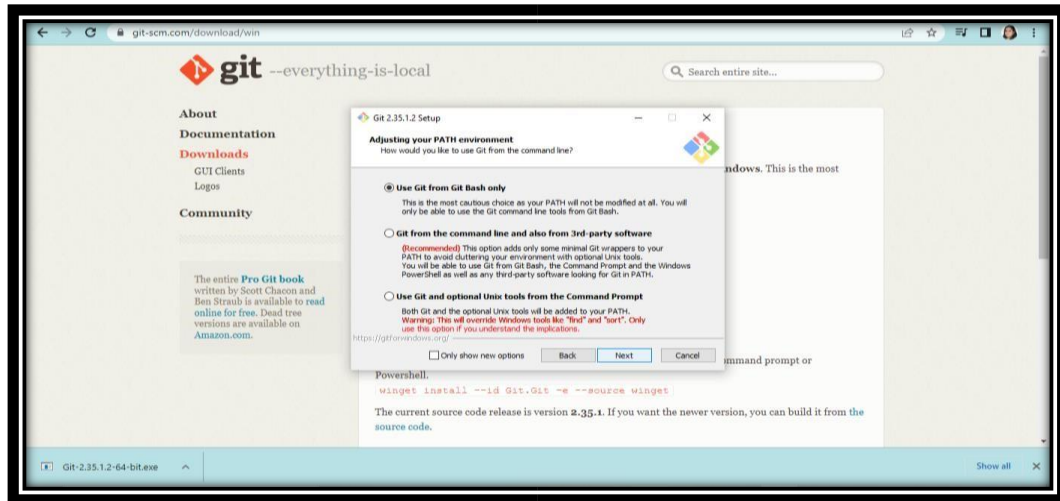
1



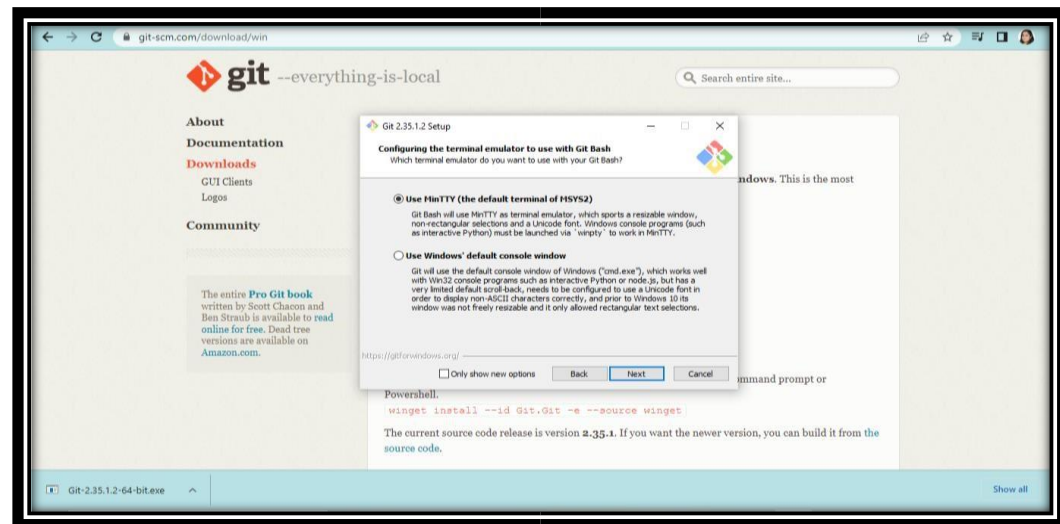
2.



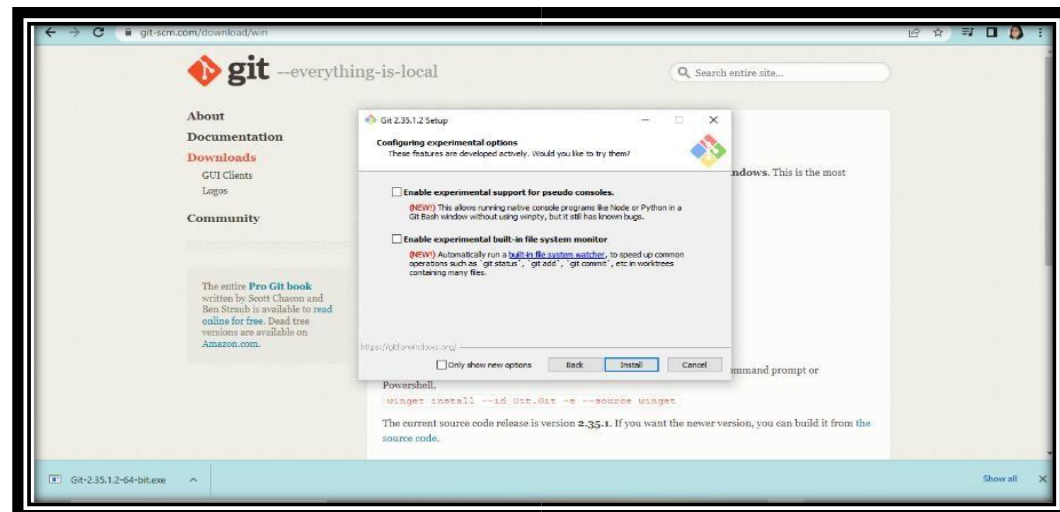
3.

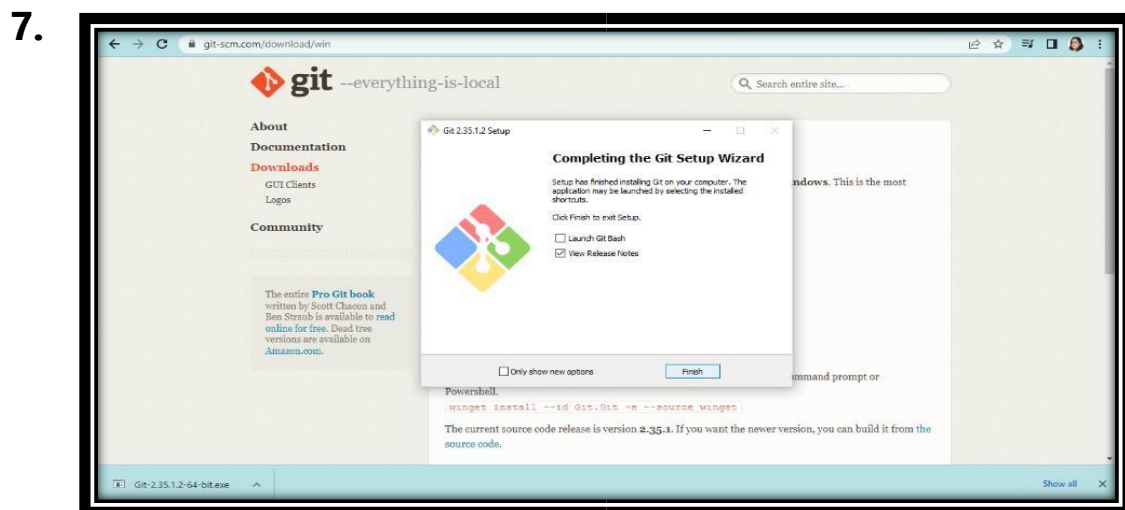
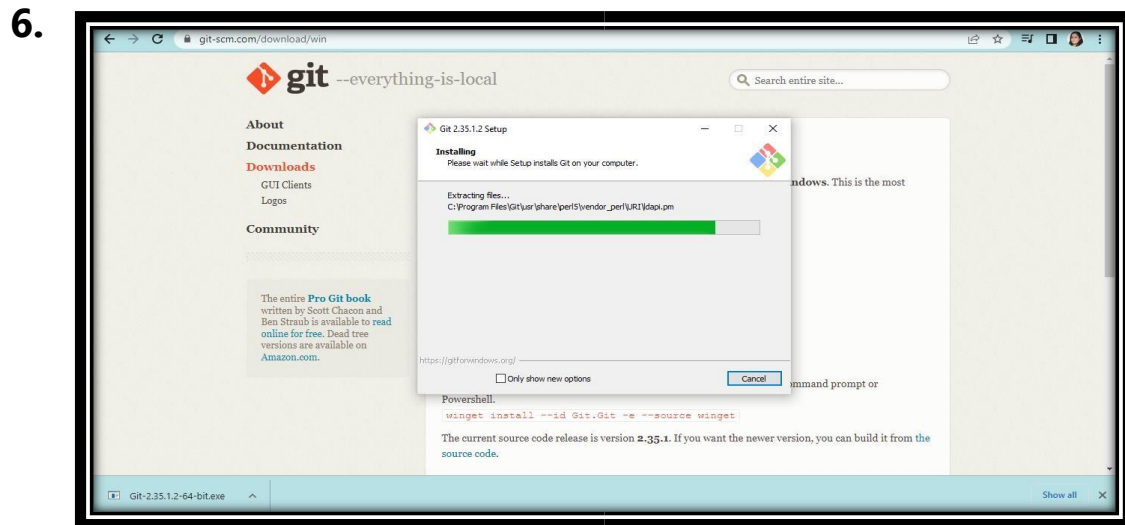


4.



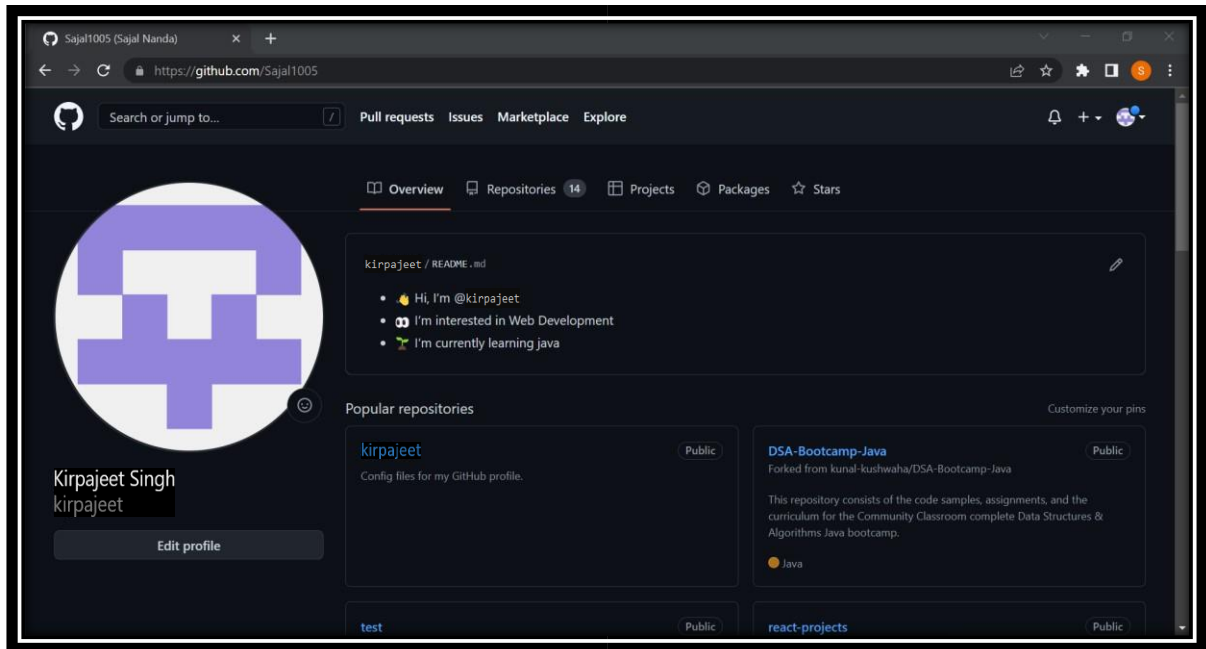
5.





## Step 3 – Setting up GitHub account:

1.

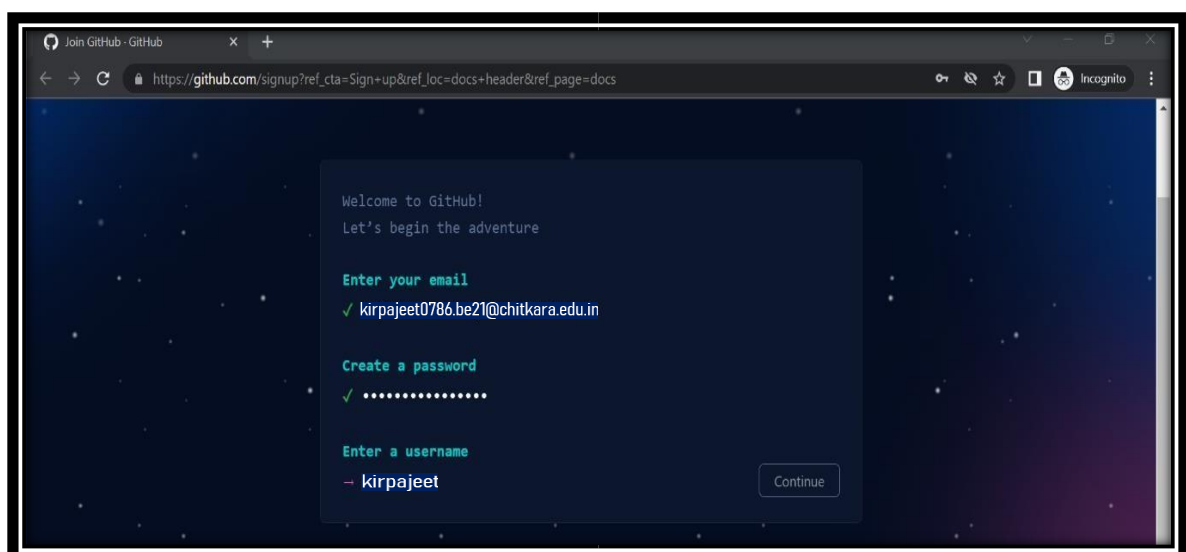


## Step1)

You can configure your Git by typing: -

1. Set your username: `git config --global user.name "Your Name"`
2. Set your email address: `git config --global user.email "Your Email"`

The page looks like as: -





```

MINGW64:/c/Users/Nightriders/Desktop/Git-Project
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (main)
$ git config --global user.name
kirpajeet
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (main)
$ git config --global user.email
kirpajeet21gill@gmail.com
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (main)
$

```

## Step2)

You can check configuration of Git by typing -

1. git config --list

The page looks like as: -

```

MINGW64:/c/Users/Nightriders/Desktop/Git-Project
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (activity1)
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Sajal1005
user.email=nandasajal.208@gmail.com
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false

```

## 4. Program to generate Logs:

### Program to Generate logs:

Advantage of version control systems like git is that it can record changes. 'Git log' command is used to display all these changes that were made by the user in the repository. Log is a record of all the previous commits.

To understand Logs we need to get familiar with all the commands that are used in making changes to a repository.

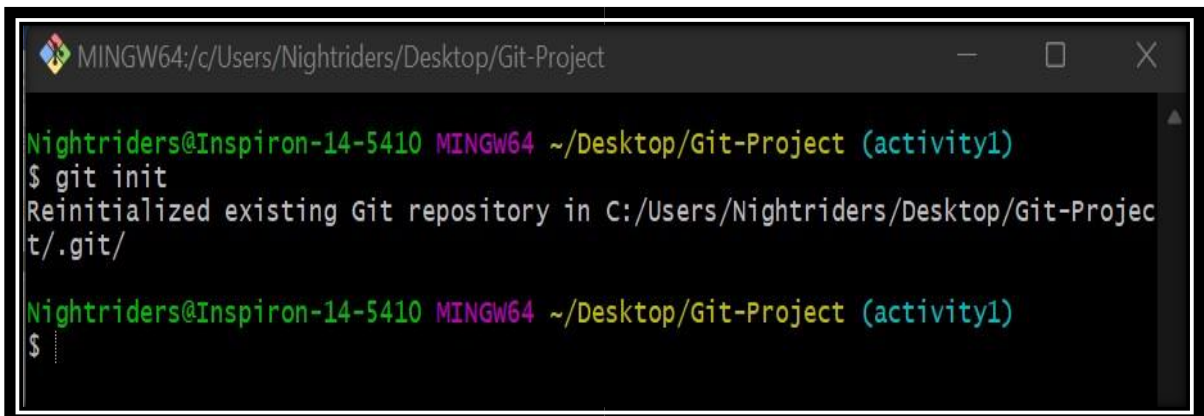
- 1) **Repository:** A repository is a directory that contains all the project-related data.
- 2) **Git init:** The git init command is used to create a new blank repository.
- 3) **Git status:** We can list all the untracked, modified and deleted files using the git status command.
- 4) **Git add:** Adds all the untracked and modified files to the staging area.
- 5) **Git commit:** Git commit finalizes the changes in the repository. Every commit is saved in the branch you are working in and can be used to revert back to older versions of the project.

## **Making GIT Repository**

### Step1: GIT INIT

Initializing a new repository, You Can do it by typing: - 1. git init

The page looks like as: -

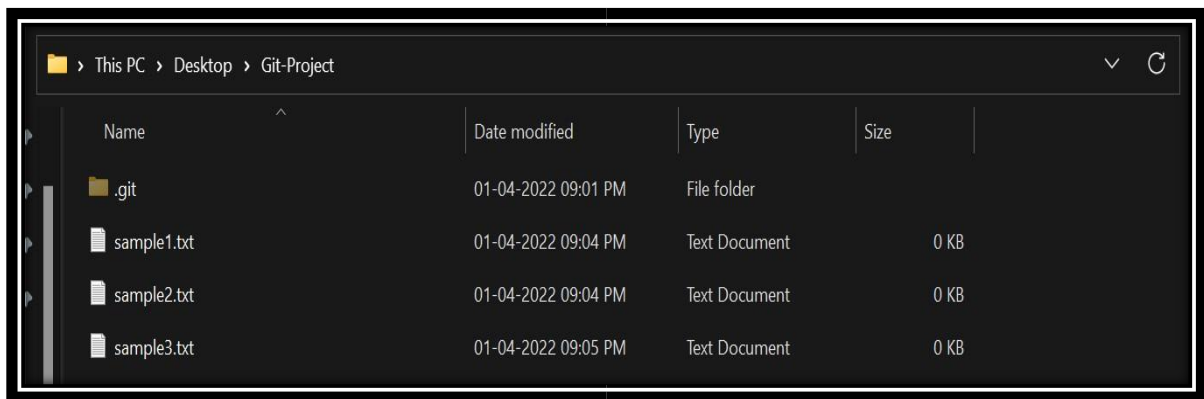


```
MINGW64:/c:/Users/Nightriders/Desktop/Git-Project
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (activity1)
$ git init
Reinitialized existing Git repository in C:/Users/Nightriders/Desktop/Git-Project/.git/
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (activity1)
$
```

### Step2: ADDING THE FILES TO THE FOLDER

Just like (Samples...)

The page looks like as: -



### Step3: GIT ADD

The git add command adds a change in the working directory to the staging area. You Can do it by typing: -

1. git add
2. git add (current file name)

The page looks like as: -

```

MINGW64:/c/Users/Nightriders/Desktop/Git-Project

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        sample1.txt
        sample2.txt
        sample3.txt

nothing added to commit but untracked files present (use "git add" to track)
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$ git add .
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$

```

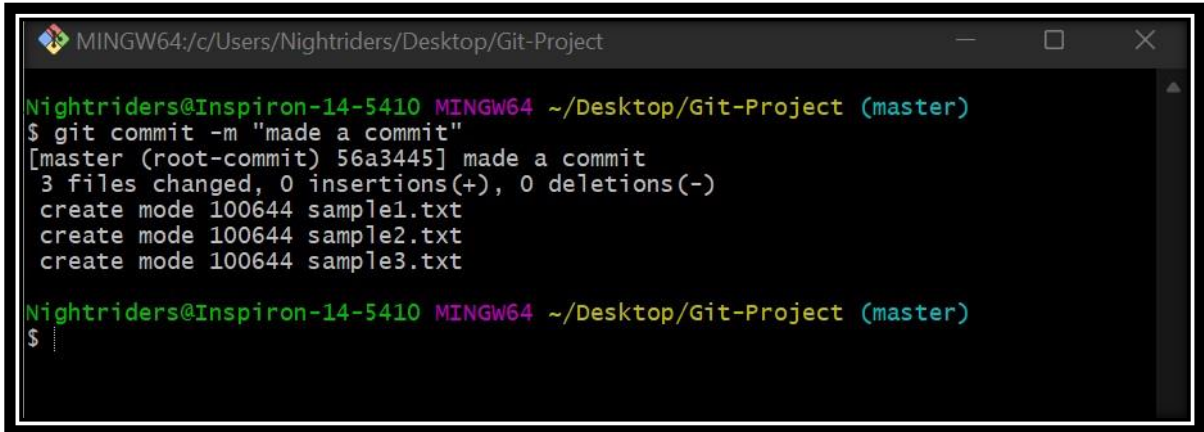
### Step4: GIT COMMIT

The "commit" command is used to save your changes to the local repository.

You Can do it by typing: -

1. git commit -m "any text"

The page looks like as: -



```
MINGW64:/c/Users/Nightriders/Desktop/Git-Project

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$ git commit -m "made a commit"
[master (root-commit) 56a3445] made a commit
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 sample1.txt
create mode 100644 sample2.txt
create mode 100644 sample3.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$
```

### Step5: GIT LOG

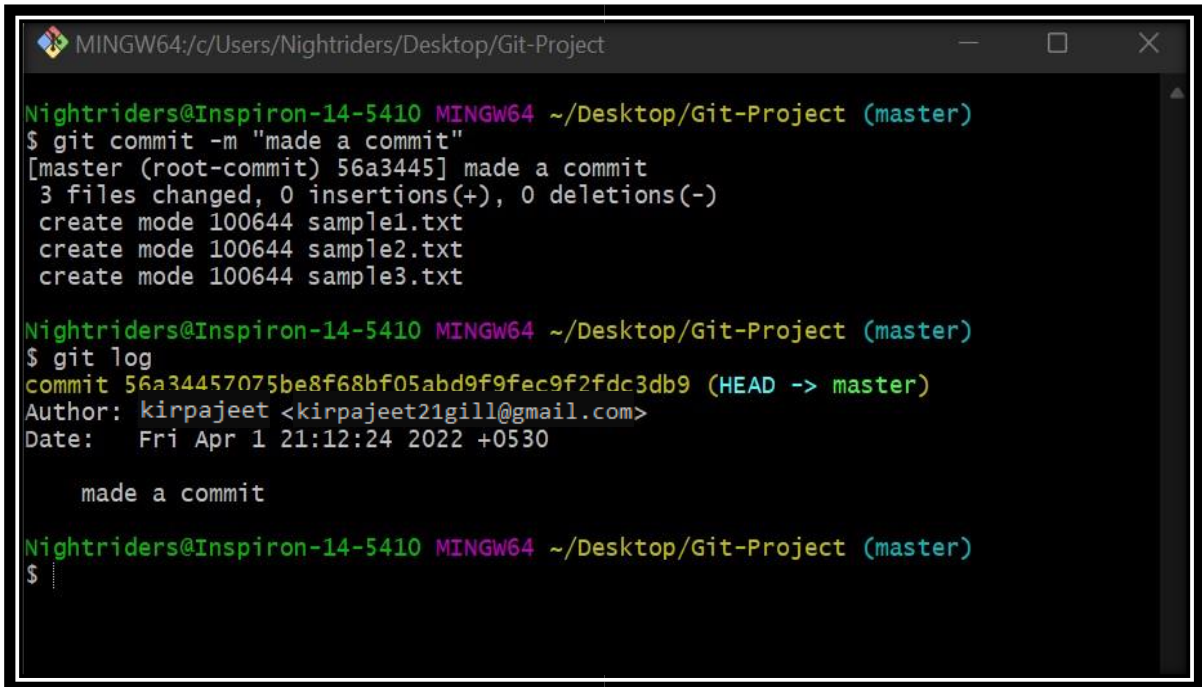
Git log will show all the commits made by the author with their time. After every commit the checksum value (written In yellow color) of the folder changes.

Checksum is used to verify that the data in that file has not been tampered with or manipulated, possibly by a malicious entity.

You Can do it by typing: -

#### 1. git log

The page looks like as: -



```

MINGW64:/c/Users/Nightriders/Desktop/Git-Project
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$ git commit -m "made a commit"
[master (root-commit) 56a3445] made a commit
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 sample1.txt
create mode 100644 sample2.txt
create mode 100644 sample3.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$ git log
commit 56a34457075be8f68bf05abd9f9fec9f2fdc3db9 (HEAD -> master)
Author: kirpajeet <kirpajeet21gill@gmail.com>
Date:   Fri Apr 1 21:12:24 2022 +0530

    made a commit

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$
  
```

## **Step6: Create and Visualize Branches**

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master.

## **5. Create and Visualize Branches**

### **Step1: CHECKING UP THE BRANCHES**

You can check which branch you are working in by using the command

1. 'git branch'.

The default branch is always the master branch.

The page looks like as: -

```

MINGW64:/c/Users/Nightriders/Desktop/Git-Project
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$ git branch sample1

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$ git branch
* master
  sample1

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$

```

## Step2: CHANGING BRANCHES

To switch to the other branch

You Can do it by typing: -

1. git checkout (BRANCH NAME)

The page looks like as: -

```

MINGW64:/c/Users/Nightriders/Desktop/Git-Project
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$ git branch
* master
  sample1

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$ git checkout sample1
Switched to branch 'sample1'

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (sample1)
$

```

## Step3: GIT MERGING

Now you can merge two branches by command.

1. git merge (BRANCH NAME)

If you want to merge a new branch in master branch you need to first checkout into the master branch and then run the command.

The page looks like as: -

```

MINGW64/c/Users/Nightriders/Desktop/Git-Project
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$ git branch
* master
  sample1

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$ git merge sample1
Updating 56a3445..44e104e
Fast-forward
 sample4.txt | 0
 sample5.txt | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 sample4.txt
 create mode 100644 sample5.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$

```

## 6. Git Lifecycle Description:

There are three stages for git lifecycle:

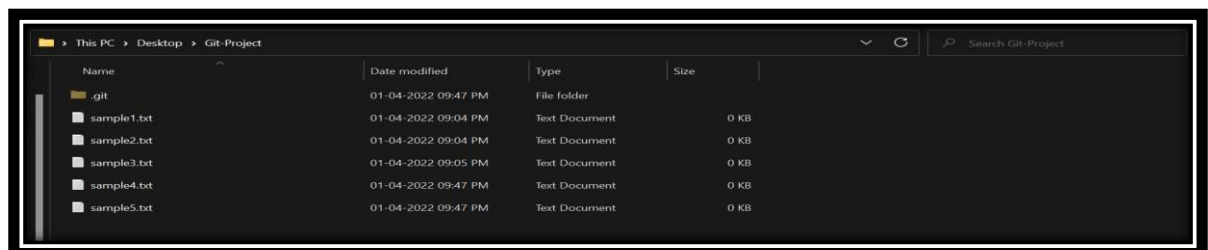
- 1) Working directory
- 2) Staging area
- 3) Git repository

### Working Directory:

The working directory is the folder in your local computer where the project files and folders are stored.

The local directory is created by the command 'git init' which creates a '.git' named folder which is used to track the files in the directory.

'git folder' is generally hidden but can be tracked enabling hidden files.



### Staging area:

The staging area has those files which are supposed to go to the next commit. Only those files which are needed to go to the next commit stay in the staging area.

You can shift the files to the git repository by using the command

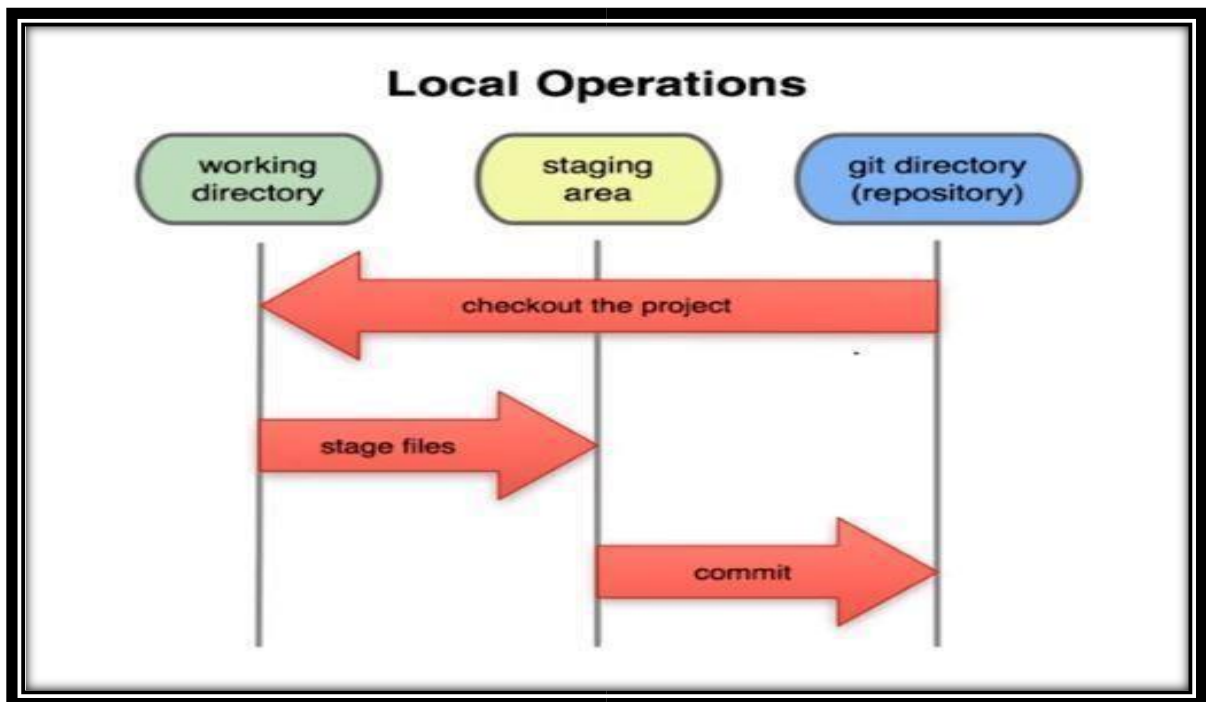


`'git add --a'.`

### Git repository:

Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the git commit command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about.

You can commit files by using command `'git commit -m "message"'`



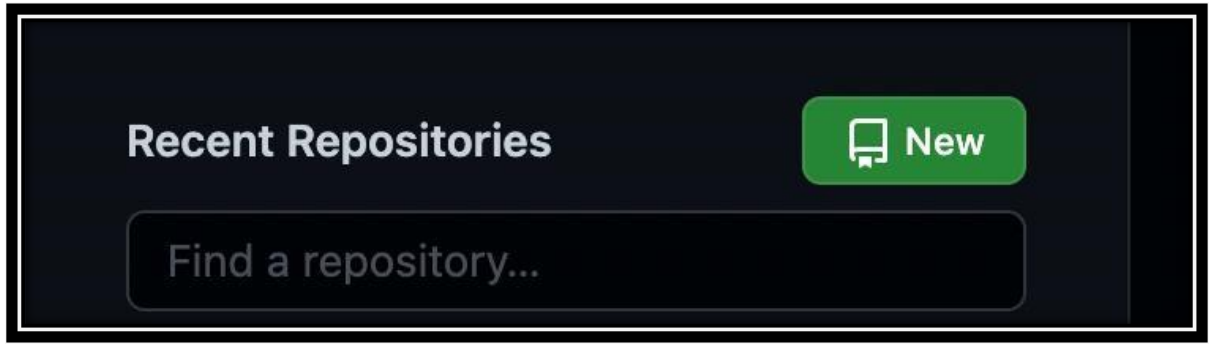
## 7. Uploading data on GitHub

**NOTE-**  
**YOU HAVE TO MAKE A REPOSITORY IN GITHUB.**

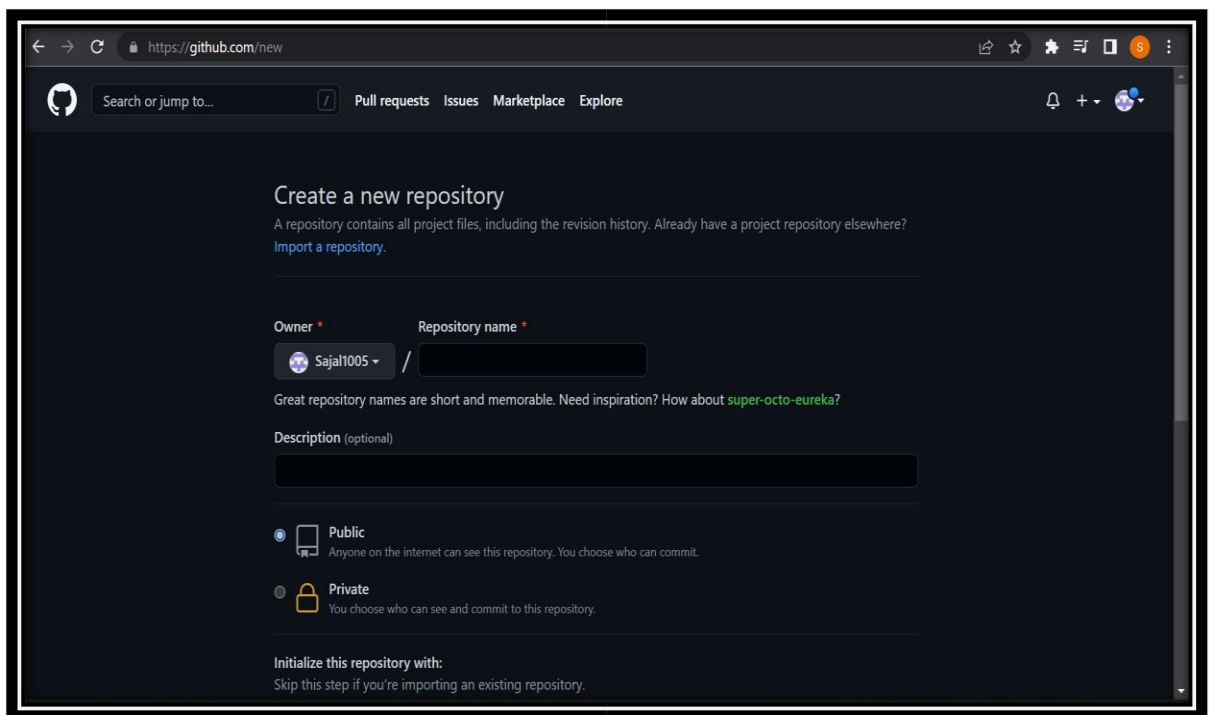


## Step1) CREATING REPOSITORY IN GITHUB

The page looks like as: -



By clicking on new you are able to make a new repository.



Write the repository name and click on next.

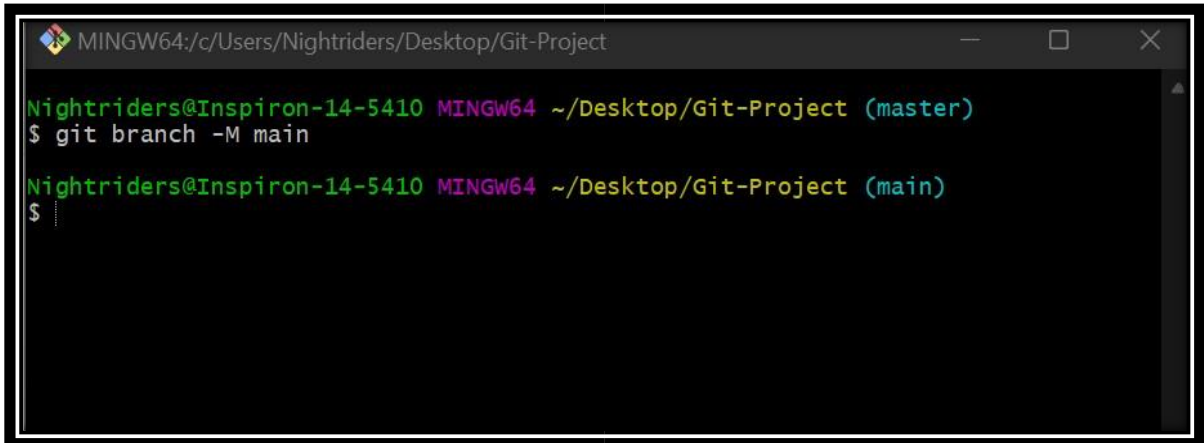


Your GITHUB Repository has been created.

## Step2) GIT ADDING REMOTE BRANCH

Git stores a branch as a reference to a commit, and a branch represents the tip of a series of commits. You Can do it by typing: - 1. `git branch -M main`

The page looks like as: -



```

MINGW64:/c/Users/Nightriders/Desktop/Git-Project
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$ git branch -M main
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (main)
$

```

## Step3) GIT ADDING REMOTE ORIGIN

Is a Git repository that's hosted on the Internet

You Can do it by typing: - 1. `git remote add origin (URL)`

The page looks like as: -



```

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (activity1)
$ git remote -v
origin https://github.com/kirpajeet/SCMProject.git (fetch)
origin https://github.com/kirpajeet/SCMProject.git (push)
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (activity1)
$

```

## Step4) GIT CLONE

The git clone command is used to upload local repository content to a remote repository.  
You Can do it by typing: -

1. `git clone url`

The page looks like as: -

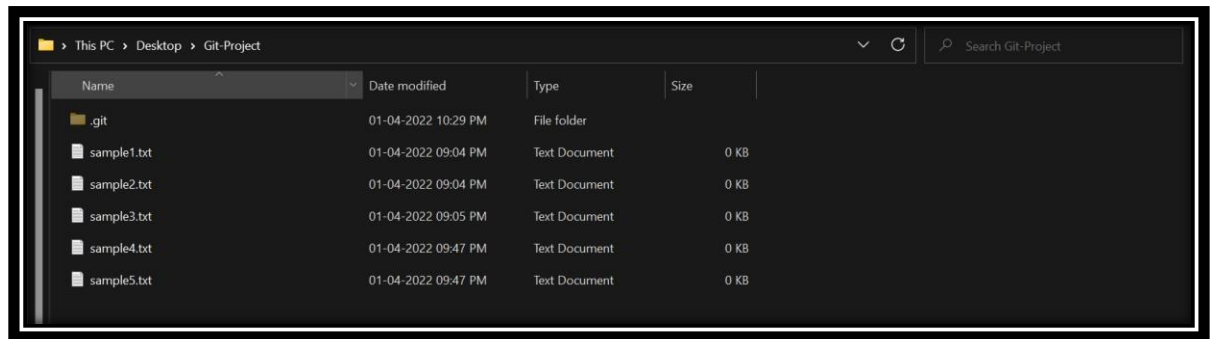
```
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (activity1)
$ git clone https://github.com/kirpajeet/SCMProject.git
Cloning into 'SCMProject'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 1), reused 5 (delta 1), pack-reused 0
Receiving objects: 100% (5/5), done.
Resolving deltas: 100% (1/1), done.

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (activity1)
$
```

## Final Result

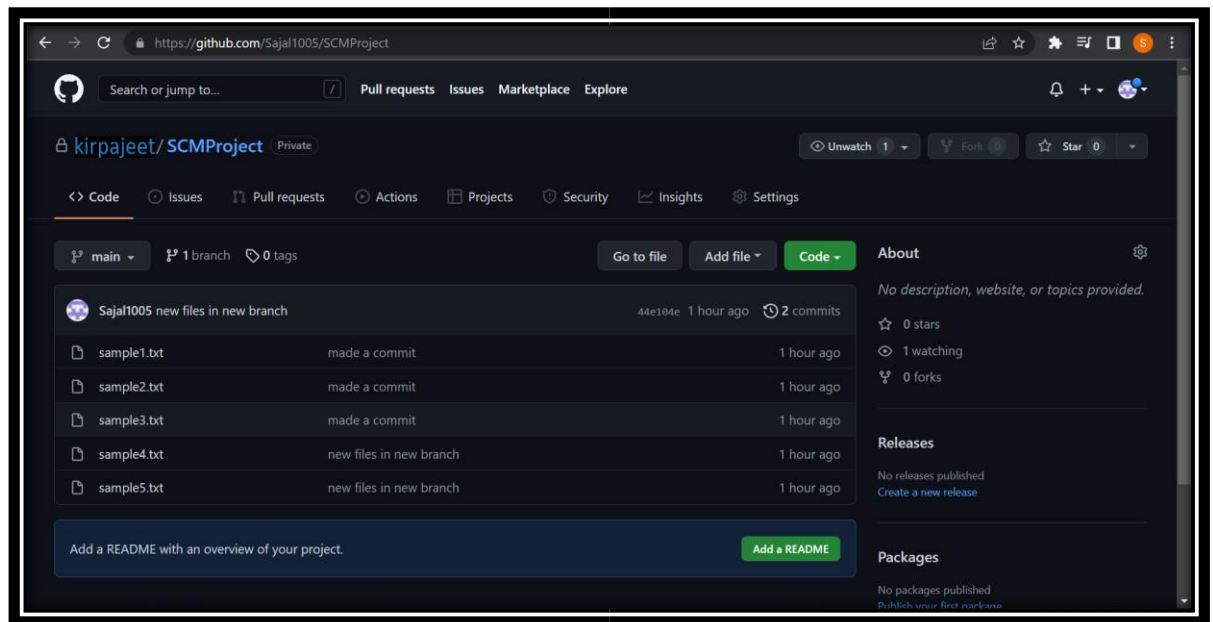
1. Document in your system

The page looks like as: -



## 2. Document in your GITHUB Repository

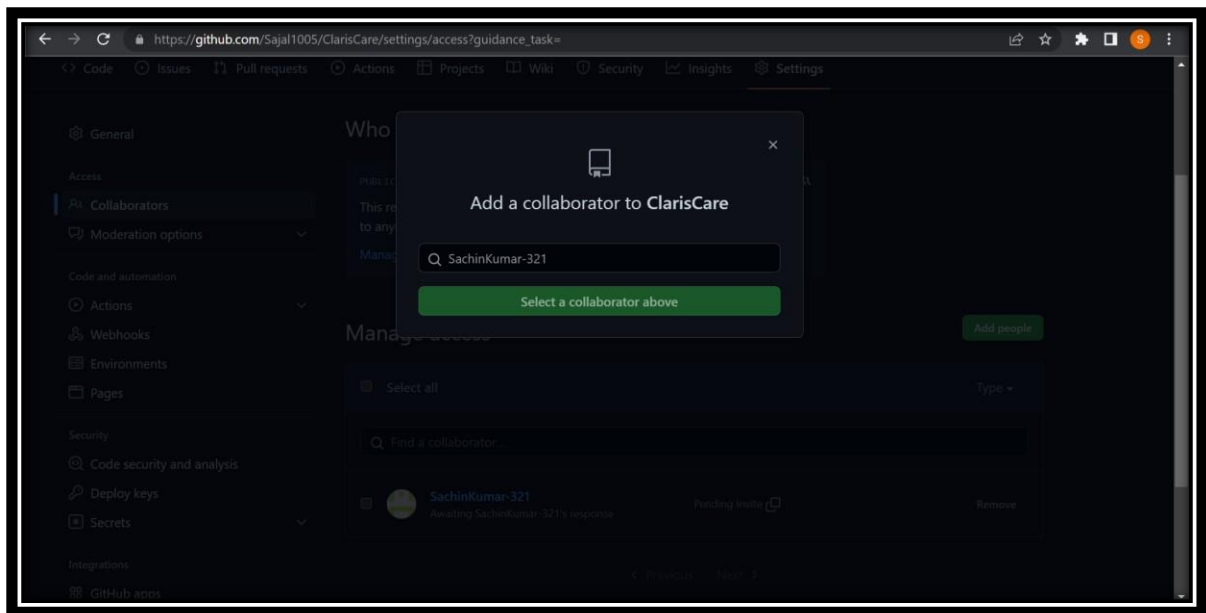
The page looks like as



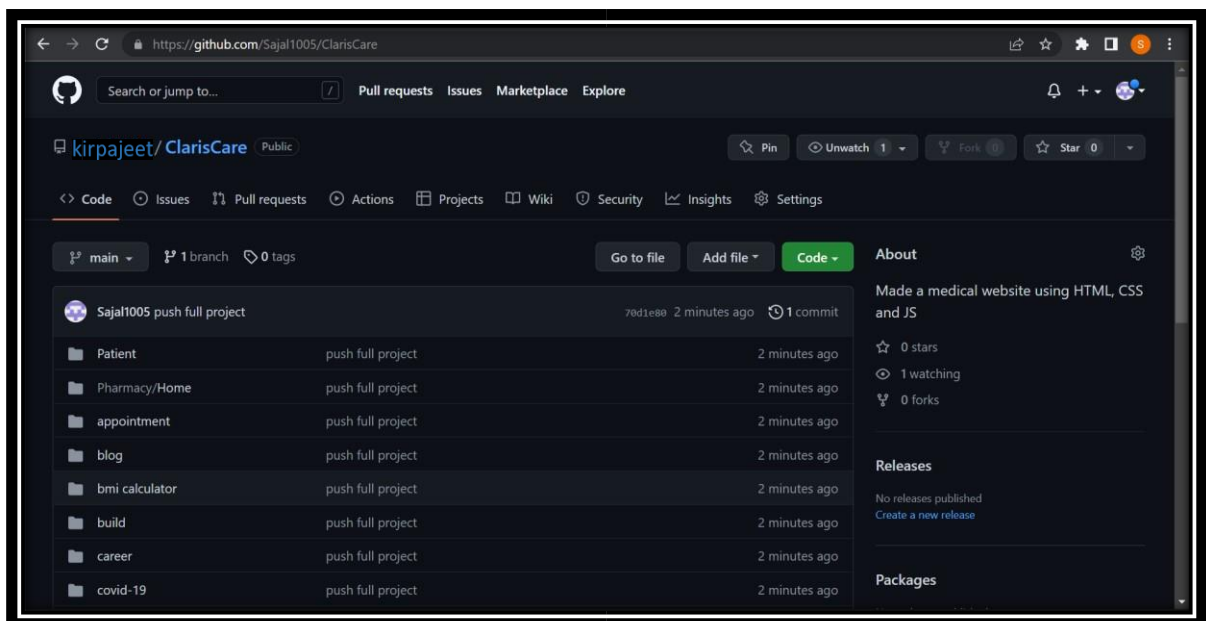
## Task 1.2

### 1. Add Collaborators on GitHub Repository: -

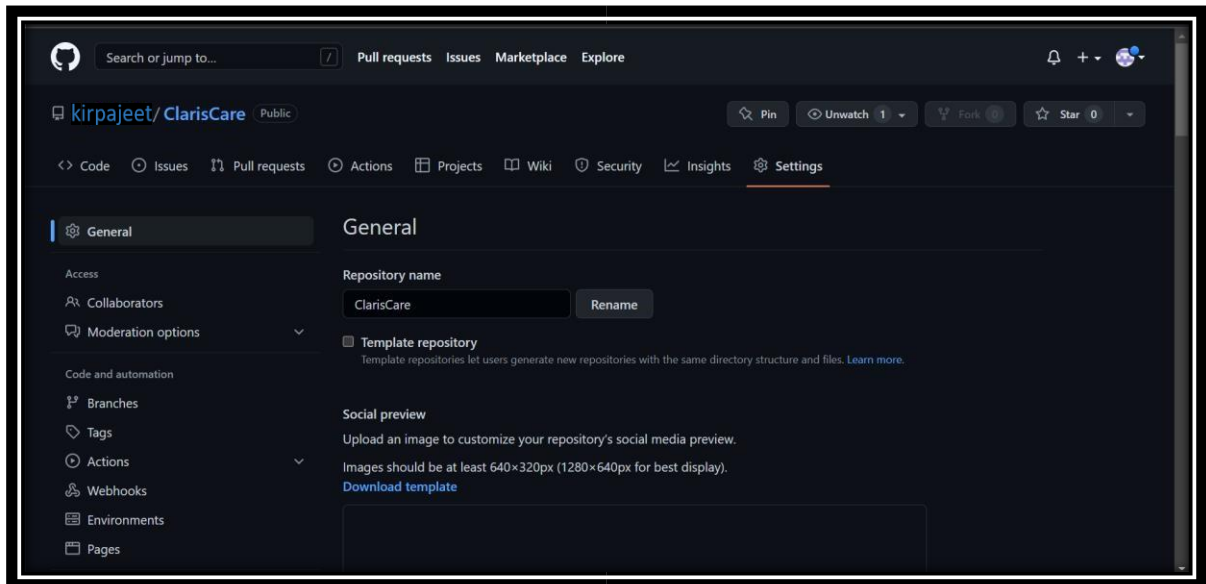
1. Ask for the username of the person you're inviting as a collaborator. ...



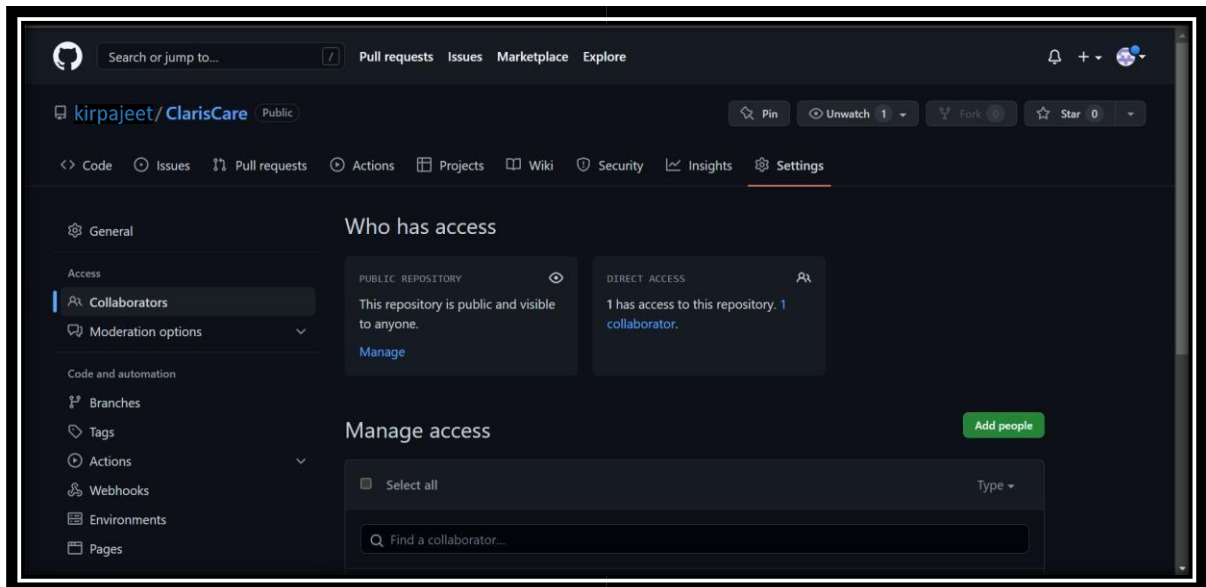
2. On GitHub.com, navigate to the main page of the repository.



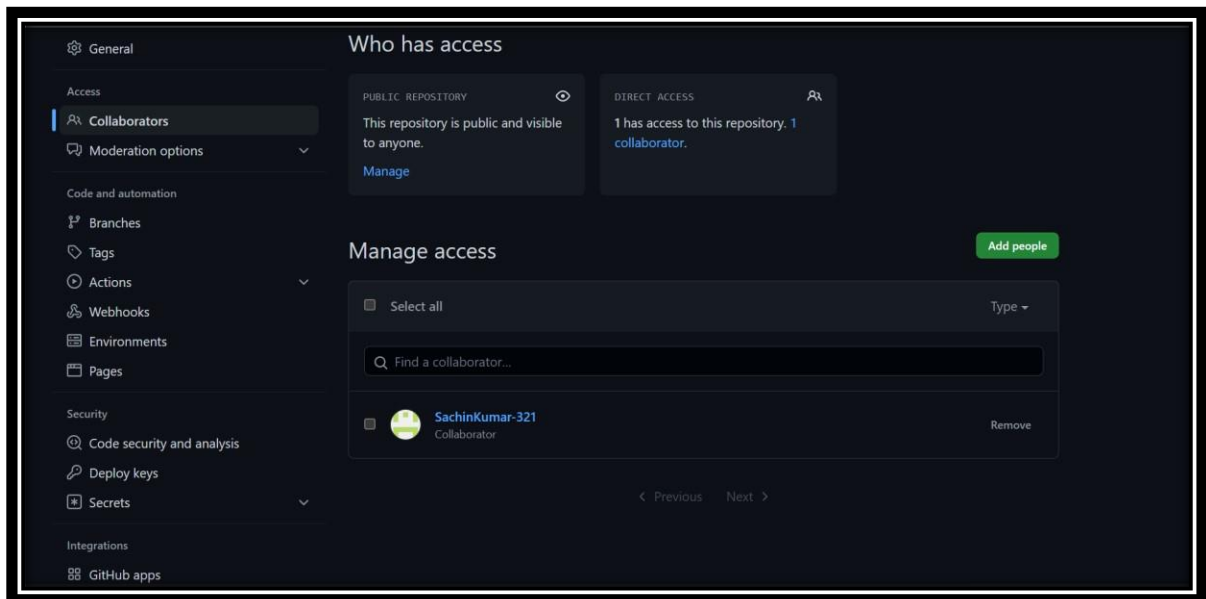
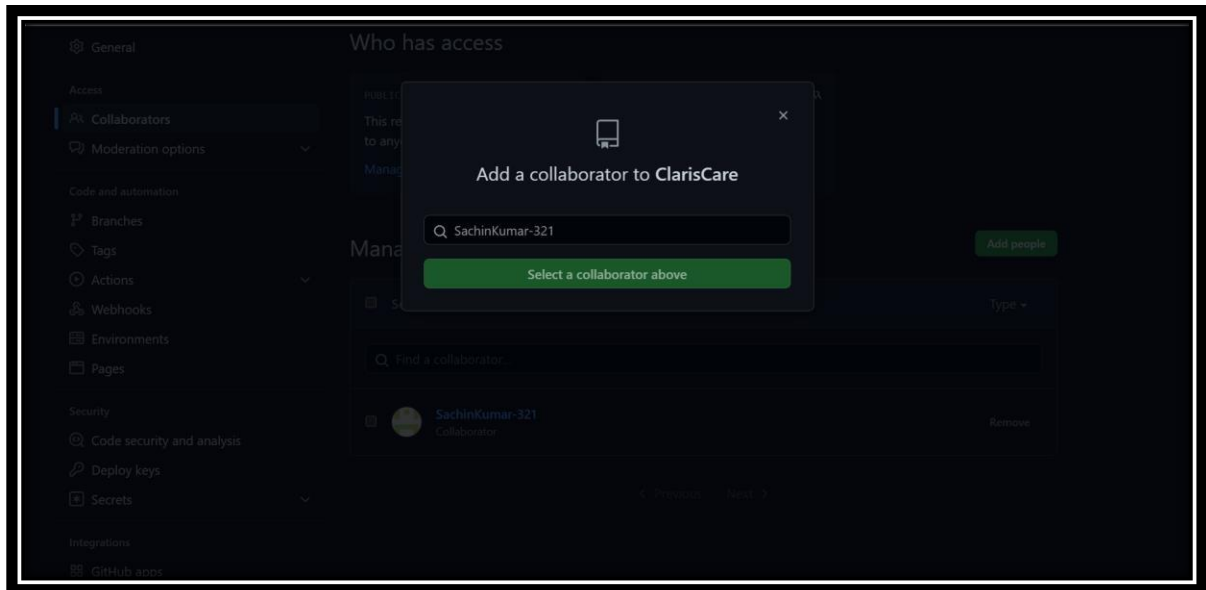
3. Under our repository name, click Settings



4. In the "Access" section of the sidebar, click Collaborators & teams.



5. Click Invite a collaborator.



**2. Fork and Commit:** - A fork is a copy of a repository that we manage. Forks let us make changes to a project without affecting the original repository. We can fetch updates from or submit changes to the original repository with pull requests.

If we need to fork a GitHub or GitLab repo, it's as simple as navigating to the landing page of the repository in your web browser and clicking on the *Fork* button on the repository's home page. A forked copy of that Git repository will be added to your personal GitHub or GitLab repo. That's it. That's all we have to do to fork a Git repo.

### **3. Merge and Resolve conflicts created due to own Activity and**

**Collaborators activity: -**

There are a few steps that could reduce the steps needed to resolve merge conflicts in Git.

1. The easiest way to resolve a conflicted file is to open it and make any necessary changes.
2. After editing the file, we can use the git add a command to stage the new merged content.
3. The final step is to create a new commit with the help of the git commit command.
4. Git will create a new merge commit to finalize the merge

Let us now look into the Git commands that may play a significant role in resolving conflicts.



```
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ ls
test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git add remote origin https://github.com/kirpajeet/SConflicts.git
fatal: pathspec 'remote' did not match any files

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git remote add origin https://github.com/kirpajeet/SConflicts.git

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ vi test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git add .
warning: LF will be replaced by CRLF in test1.txt.
The file will have its original line endings in your working directory

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git add .

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$
```

```
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git commit -m"first commit"
[main (root-commit) b6f7ef5] first commit
1 file changed, 2 insertions(+)
create mode 100644 test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git push -u origin master
error: src refspec master does not match any
error: failed to push some refs to 'https://github.com/Sajal1005/Conflicts.git'

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 266 bytes | 266.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Sajal1005/Conflicts.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git branch test

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git checkout test
Switched to branch 'test'

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (test)
$ vi test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (test)
$ git add .
warning: LF will be replaced by CRLF in test1.txt.
The file will have its original line endings in your working directory

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (test)
$ git add .

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (test)
$ git commit -m"second commit"
[test 4eb058a] second commit
1 file changed, 2 insertions(+), 1 deletion(-)
```

```
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (test)
$ git push -u origin test
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 327 bytes | 327.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'test' on GitHub by visiting:
remote:   https://github.com/kirpajeet/Sonflicts/pull/new/test
remote:
To https://github.com/kirpajeet/Sonflicts.git
 * [new branch]      test -> test
Branch 'test' set up to track remote branch 'test' from 'origin'.

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (test)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git checkout master
error: pathspec 'master' did not match any file(s) known to git

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ vi test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git add .

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git commit -m"third commit"
[main 0bc94ca] third commit
1 file changed, 2 insertions(+), 1 deletion(-)

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
```

```
Merge branch 'main' of https://github.com/kirpajeet/Sonflicts into test

# Conflicts:
#   test1.txt
#
# It looks like you may be committing a merge.
# If this is not correct, please run
#   git update-ref -d MERGE_HEAD
# and try again.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch test
# Your branch is up to date with 'origin/test'.
#
# All conflicts fixed but you are still merging.
#
# Changes to be committed:
#   modified:   test1.txt
#
~
~
~
```

```
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (main)
$ git checkout test
Switched to branch 'test'
Your branch is up to date with 'origin/test'.

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (test)
$ git pull origin main
From https://github.com/kirpajeet/Sonflicts
[test d9352ce] Merge branch 'main' of https://github.com/kirpajeet/Sonflicts into te

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Conflict (test)
$ git push -u origin test
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 477 bytes | 477.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Sajal1005/Conflicts.git
 4eb058a..d9352ce test -> test
Branch 'test' set up to track remote branch 'test' from 'origin'.
```

**4. Git Reset:** - Git reset is a powerful command that is used to undo local changes to the state of a git repo. Git reset operates on "The Three Trees of Git". These trees are the Commit History (HEAD), the Staging Index, and the Working Directory.

The easiest way to undo the last Git commit is to execute the “git reset” command with the “--soft” option that will preserve changes done to your files.

Git reset --hard, which will completely destroy any changes and remove them from the local directory.



```
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git init
Initialized empty Git repository in C:/Users/Nightriders/Desktop/Reset/.git/

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ touch test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ vi test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git add test1.txt
warning: LF will be replaced by CRLF in test1.txt.
The file will have its original line endings in your working directory

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git add test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git commit -m"first commit"
[master (root-commit) 93aceb6] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git status
On branch master
nothing to commit, working tree clean

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$
```

```
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git reset --soft Head~1
fatal: ambiguous argument 'Head~1': unknown revision or path not in the working tree
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ vi test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git add .
warning: LF will be replaced by CRLF in test1.txt.
The file will have its original line endings in your working directory

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git add .

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git commit -m"second commit"
[master 60b2fde] second commit
 1 file changed, 1 insertion(+)

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git status
On branch master
nothing to commit, working tree clean

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git reset --soft Head~1

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test1.txt
```

```
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ notepad test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git add .
warning: LF will be replaced by CRLF in test1.txt.
The file will have its original line endings in your working directory

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git add .

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git commit -m"second commit"
[master f4baa2e] second commit
1 file changed, 2 insertions(+)

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git reset --hard Head~1
HEAD is now at 93aceb6 first commit

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ notepad test1.txt

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ touch test2.bin

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ touch .gitignore

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ vi test2.bin

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        test2.bin

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

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git add .
warning: LF will be replaced by CRLF in test2.bin.
The file will have its original line endings in your working directory
```

```
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git commit -m"third commit"
[master 16fe491] third commit
2 files changed, 1 insertion(+)
create mode 100644 .gitignore
create mode 100644 test2.bin

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ git status
On branch master
nothing to commit, working tree clean

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Reset (master)
$ vi test2.bin
```