



**git**

**CHITKARA**  
UNIVERSITY



# SOURCE CODE MANAGEMENT FILE

**Submitted by:**

**Name:** Navjinder Singh

**Roll No.:** 2310991996

**Group:** 22-B

**Submitted To:**

Dr. Sharad Chauhan

Professor,

CSE, Chitkara University

Rajpura, Punjab

**Submission of:** Task1.1

**Subject Name:** Source  
Code Management

**Sub Code:** 22CS003

**Department:** CSE



## **Source Code Management File**

Subject Name: **Source Code Management (SCM)**

Subject Code: **22CS003**

### **Submitted By:**

Name: **Navjinder Singh**

Roll No. **2310991996**

Group: **G22-B**

### **Task 1.1 Submission (Week 4)**

1. Setting up of Git Client
2. Setting up GitHub Account
3. Generate logs
4. Create and visualize branches
5. Git lifecycle description

# EXPERIMENT - 1

**Aim:** Setting up of Git Client

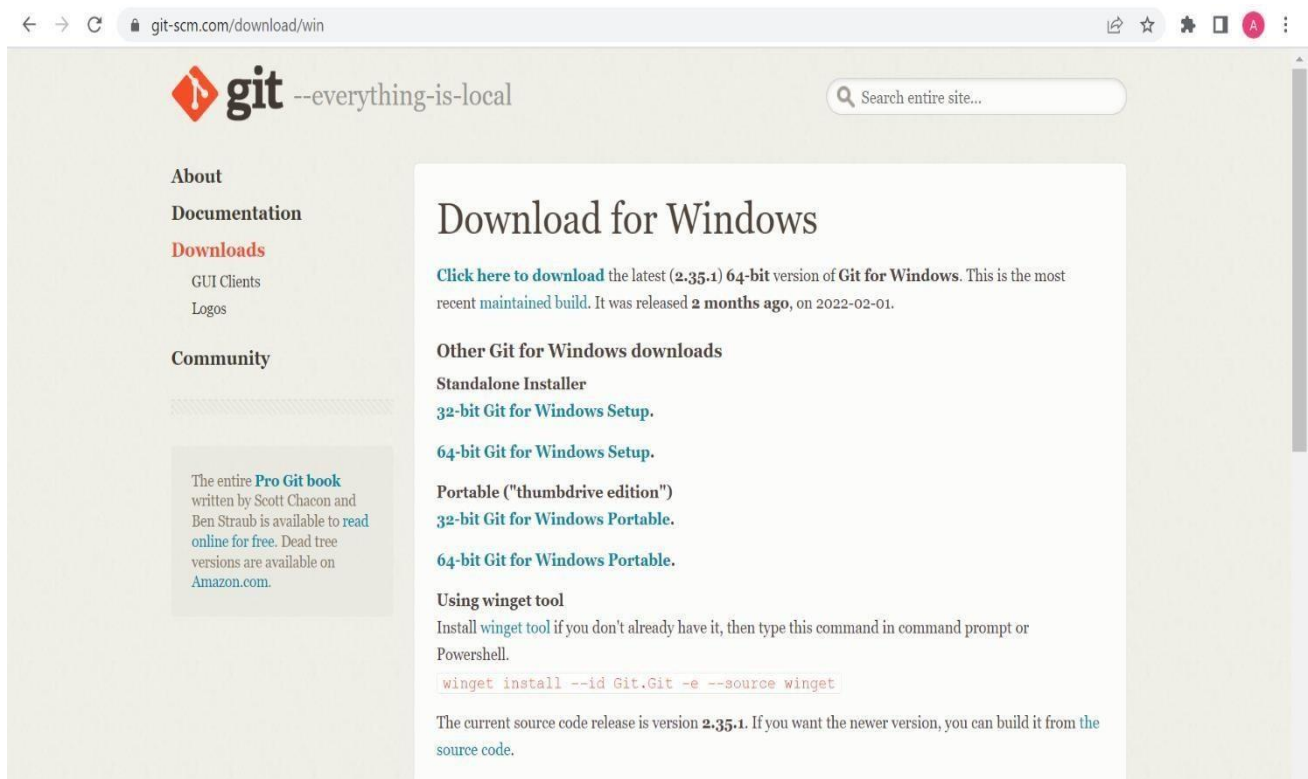
## Theory:

**GIT:** It's a Version Control System (VCS). It is a software or we can say a server by which we are able to track all the previous changes in the code. It is basically used for pushing and pulling of code. We can use git and git-hub parallelly to work with multiple members or individually. We can make, edit, recreate, copy or download any code on git hub using git.






## Procedure:

We can install Git on Windows, using the most official build which is available for download on the GIT's official website or by just typing (scm git) on any search engine. We can go on <https://git-scm.com/download/win> and can select the platform and bit-version to download. And after clicking on your desired bit-version or ios it will start downloading automatically.

## Snapshots of download:



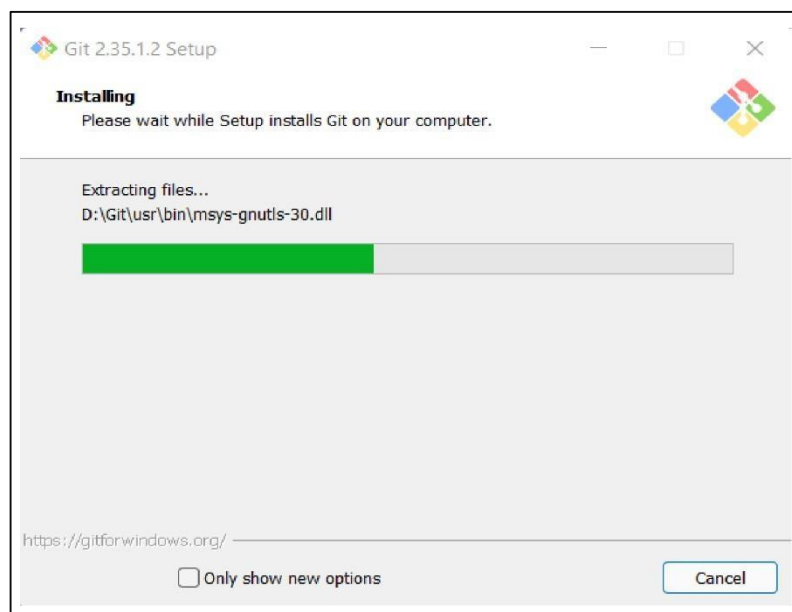
Opted for “64-bit Git for Windows Setup”

Name	Date modified	Type	Size
 Git Bash	16-03-2022 08:51	Shortcut	2 KB
 Git CMD	16-03-2022 08:51	Shortcut	2 KB
 Git FAQs (Frequently Asked Questions)	16-03-2022 08:51	Internet Shortcut	1 KB
 Git GUI	16-03-2022 08:51	Shortcut	2 KB
 Git Release Notes	16-03-2022 08:51	Shortcut	2 KB

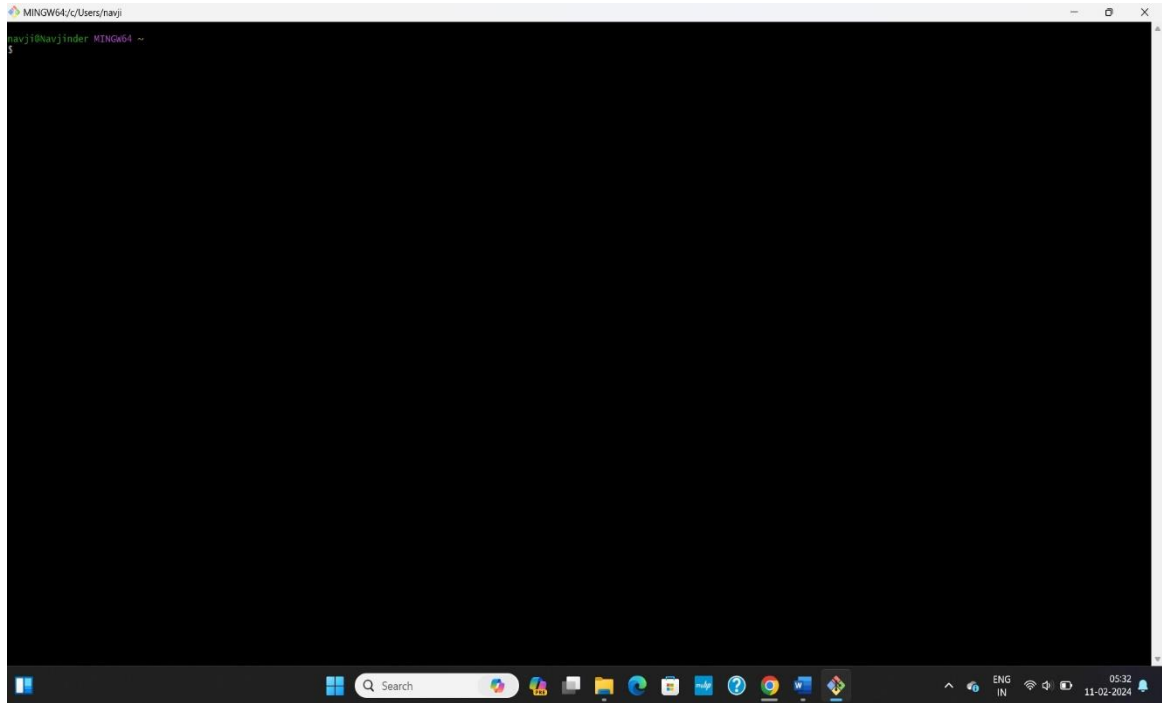
## Git and its files in downloads



## Git Setup



## Git Installation



Git Bash launched



# EXPERIMENT - 2

**Aim:** Setting up GitHub Account

## Theory:

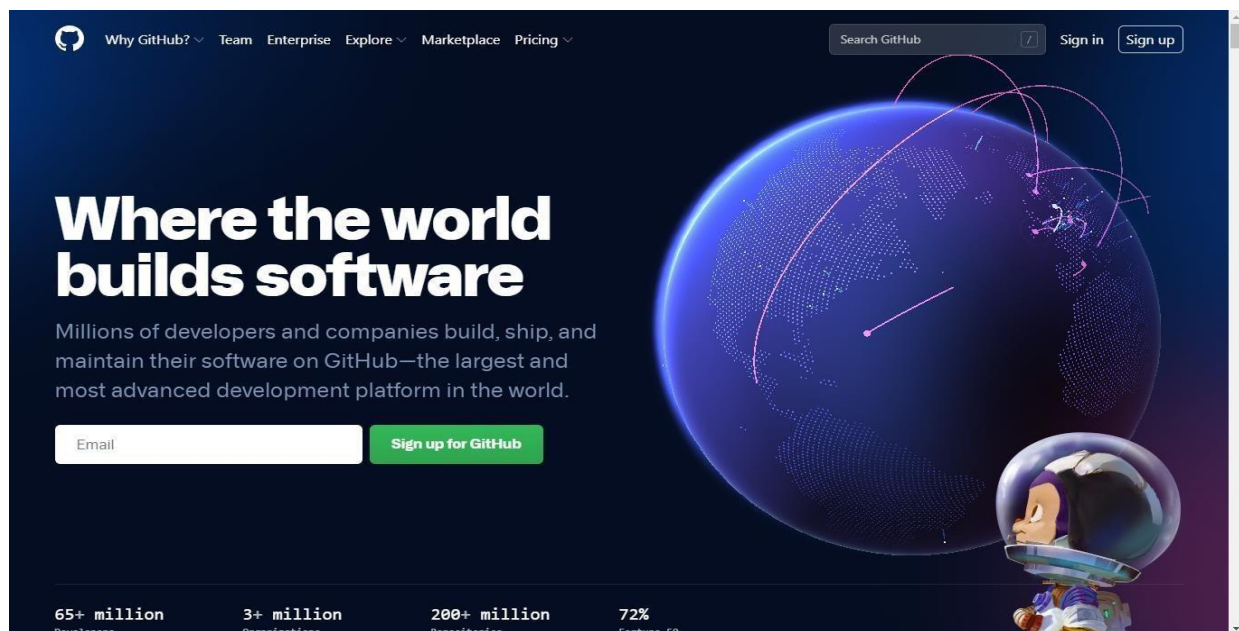
**GitHub:** GitHub is a website and cloud-based service (client) that helps an individual or developers to store and manage their code. We can also track as well as control changes to our or public code.

**Advantages of GitHub:** GitHub has a user-friendly interface and is easy to use. We can connect the git-hub and git but using some commands shown below in figure 001. Without GitHub we cannot use Git because it generally requires a host and if we are working for a project, we need to share it with our team members, which can only be done by making a repository. Additionally, anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects.

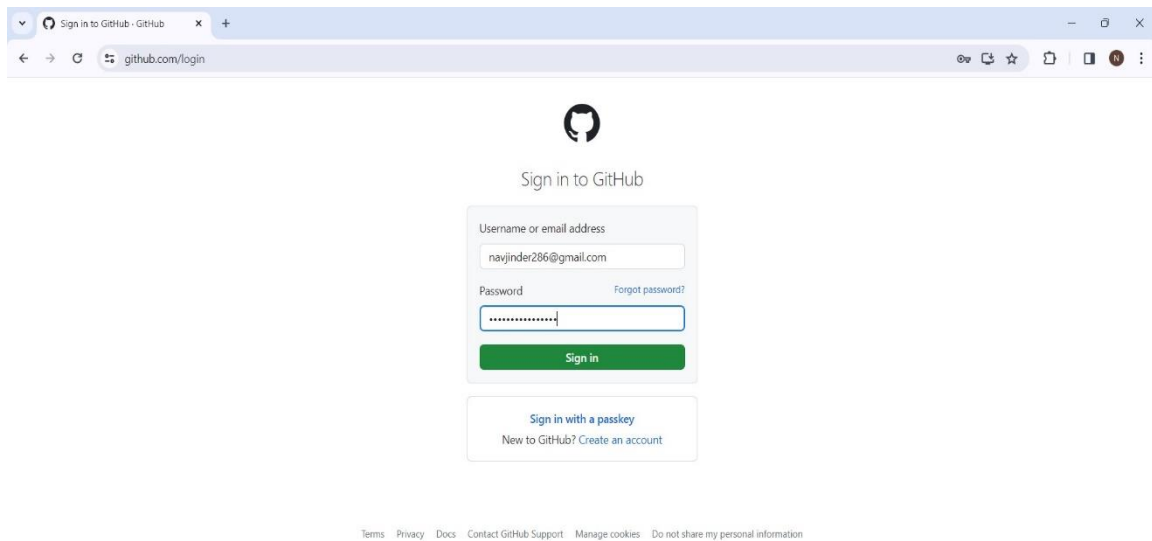
## Procedure:

To make an account on GitHub, we search for GitHub on our browser or visit <https://github.com/signup>. Then, we will enter our mail ID and create a username and password for a GitHub account.

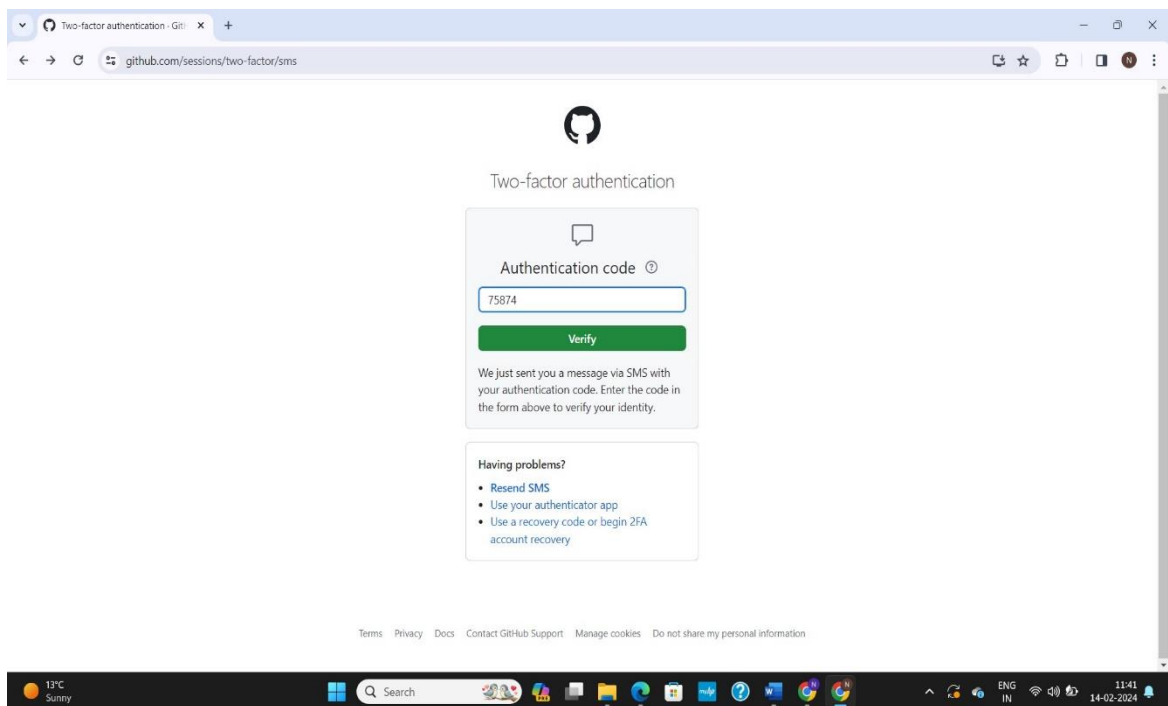
## Snapshots :



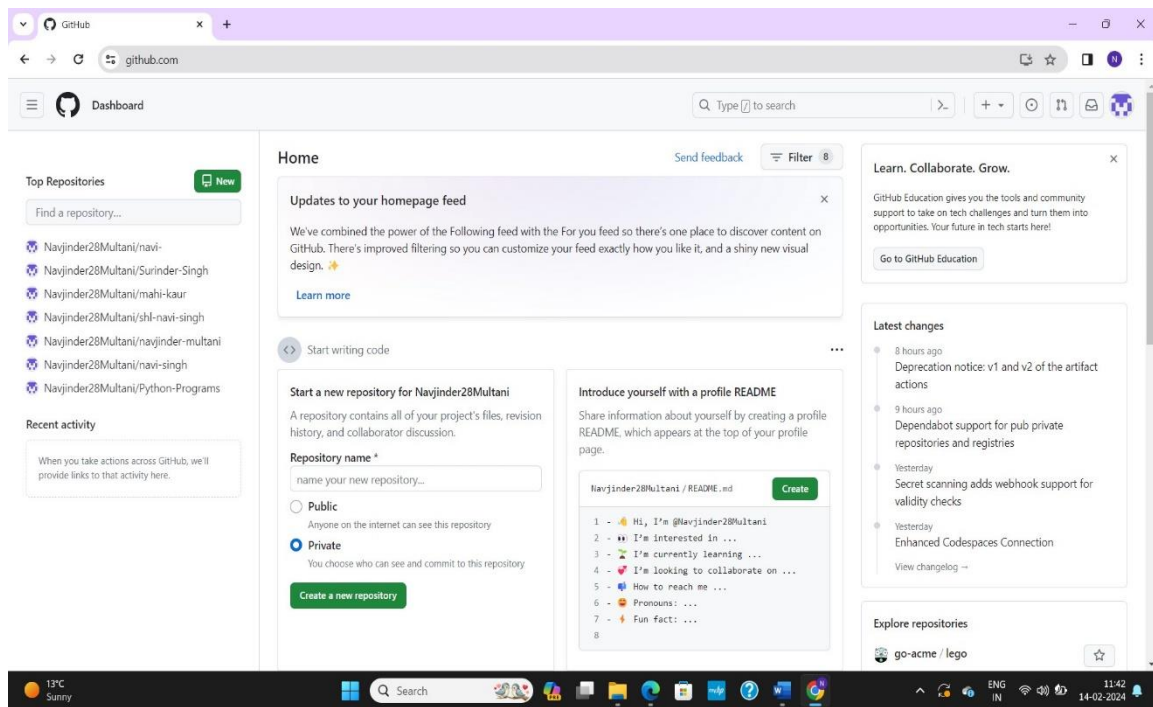
After visiting the link this type of interface will appear, if you already have an account, you can sign in and if not, you can create.



## GitHub Login



## Two Factor Authentication



GitHub Interface



# EXPERIMENT - 3

**Aim:** Program to Generate log

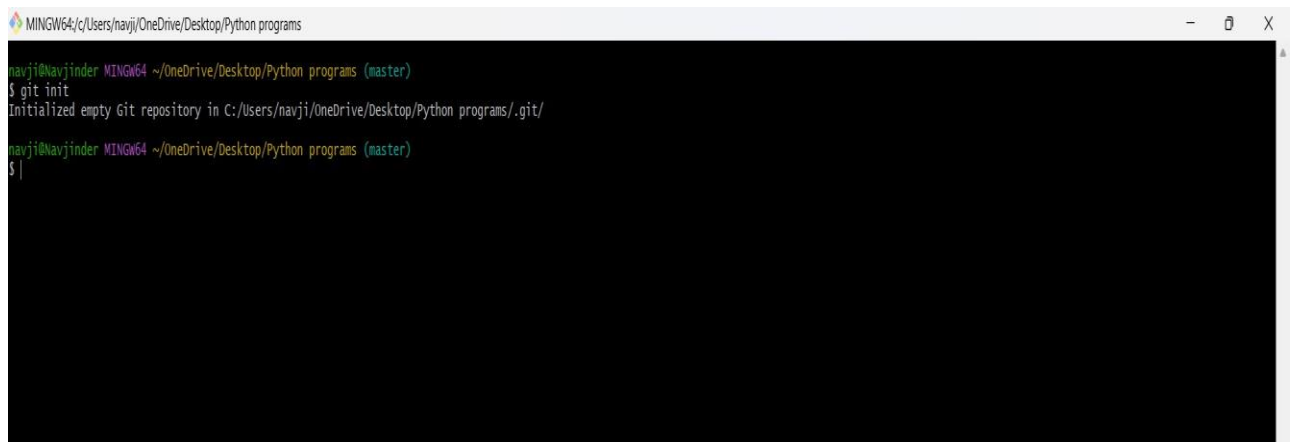
## Theory:

**Logs:** Logs are nothing but the history which we can see in git by using the code git log.

It contains all the past commits, insertions and deletions in it which we can see any time. Logs helps to check that what were the changes in the code or any other file and by whom. It also contains the number of insertions and deletions including at which time it was changed.

## Procedure:

First of all, create a local repository using Git. For this, you have to make a folder in your device, right click and select “Git Bash Here”. This opens the Git terminal. To create a new local repository, use the command “git init” and it creates a hidden folder “.git”.



```
MINGW64/c/Users/navji/OneDrive/Desktop/Python programs
navji@Navjinder MINGW64 ~/OneDrive/Desktop/Python programs (master)
$ git init
Initialized empty Git repository in C:/Users/navji/OneDrive/Desktop/Python programs/.git/
navji@Navjinder MINGW64 ~/OneDrive/Desktop/Python programs (master)
$
```

When we use GIT for the first time, we have to give the user name and email so that if I am going to change in project, it will be visible to all.

For this, we use command:

“git config --global user.name Name” “git config --global user.email email”

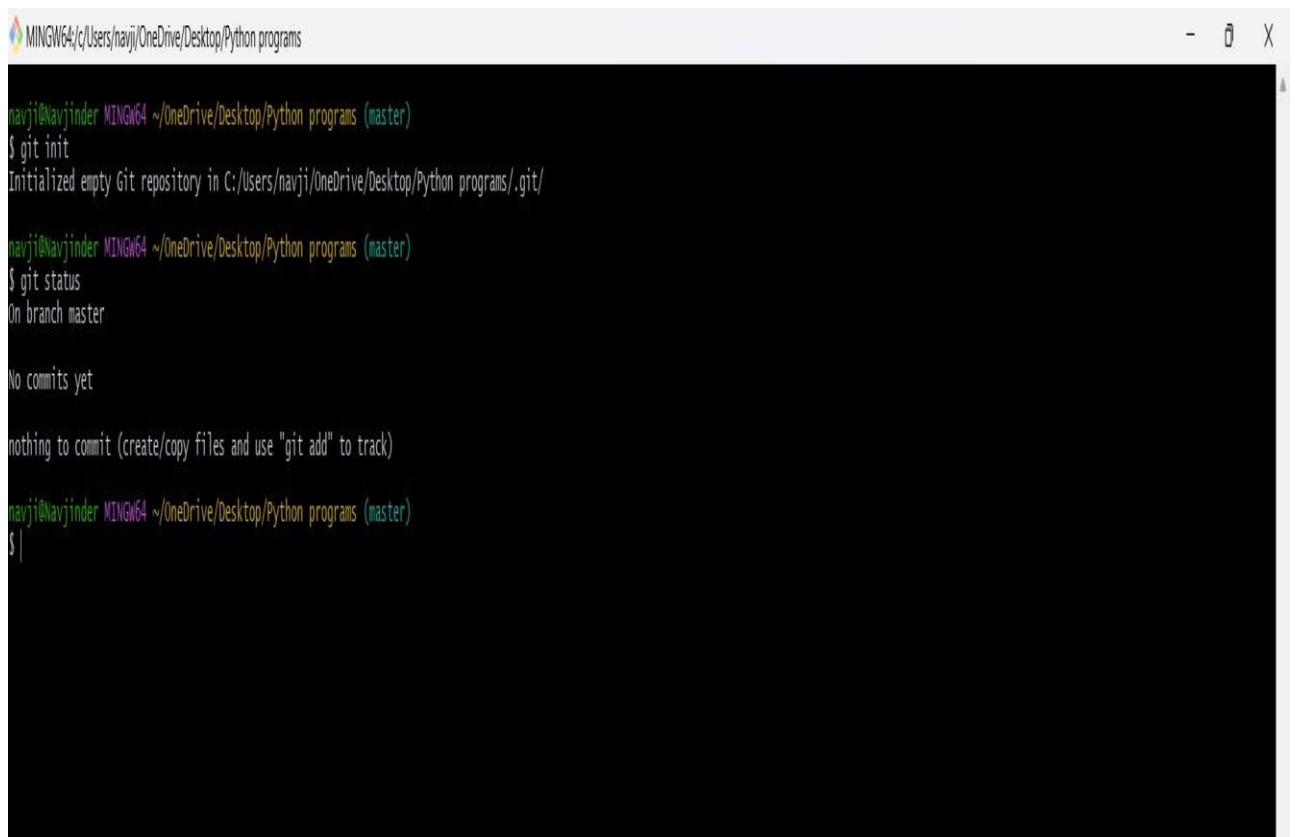
For verifying the user’s name and email, we use: “git

config --global user.name”  
“git config --global user.email”

## Some Important Commands

- `ls` - It gives the file names in the folder.
- `ls -lart` - Gives the hidden files also.
- `git status` - Displays the state of the working directory and the staged snapshot.
- `touch filename` - This command creates a new file in the repository.
- `clear` - It clears the terminal.
- `rm -rf .git` - It removes the repository.
- `git log` - displays all of the commits in a repository's history
- `git diff` - It compares my working tree to staging area.

### Git status:

A screenshot of a Windows terminal window with a dark background and light-colored text. The window title bar shows the path 'MINGW64;C:/Users/navji/OneDrive/Desktop/Python programs'. The terminal content shows the following sequence of commands and output:

```
navji@Navjinder MINGW64 ~/OneDrive/Desktop/Python programs (master)
$ git init
Initialized empty Git repository in C:/Users/navji/OneDrive/Desktop/Python programs/.git/

navji@Navjinder MINGW64 ~/OneDrive/Desktop/Python programs (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

navji@Navjinder MINGW64 ~/OneDrive/Desktop/Python programs (master)
$ |
```

```
MINGW64~/Users/navji/OneDrive/Desktop
navji@Navjinder MINGW64 ~/OneDrive/Desktop (master)
$ git init
Reinitialized existing Git repository in C:/Users/navji/OneDrive/Desktop/.git/

navji@Navjinder MINGW64 ~/OneDrive/Desktop (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        desktop.ini
        navi singh/

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

navji@Navjinder MINGW64 ~/OneDrive/Desktop (master)
$ git add .
```

Git log :

```
MINGW64~/Users/navji/OneDrive/Desktop/navi singh
navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git commit -m "This is my First Git Commit"
[master (root-commit) 6b418c6] This is my First Git Commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 master.txt

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git status
On branch master
nothing to commit, working tree clean

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git log
commit 6b418c61c4b860eebd1a81eadf071b7fa9f6b56d (HEAD -> master)
Author: Navjinder Singh <navjinder286@gmail.com>
Date: Mon Feb 12 12:36:58 2024 +0530

    This is my First Git Commit

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ |
```

The git log command displays a record of the commits in a Git repository. By default, the git log command displays a commit hash, the commit message and other commit metadata.

## EXPERIMENT - 4

**Aim:** Create and visualize branches

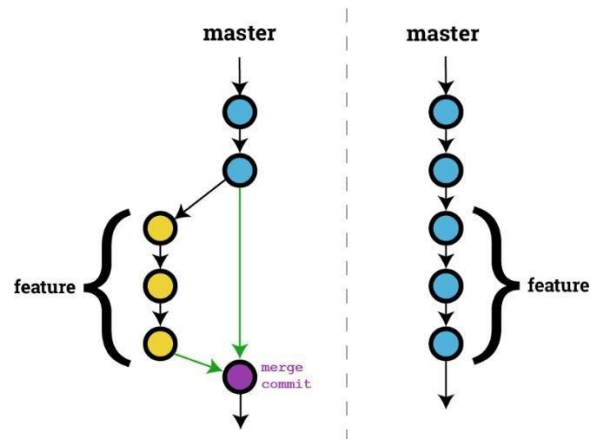
### Theory:

**Branching:** A branch in Git is an independent line of work (a pointer to a specific commit). It allows users to create a branch from the original code (master branch) and isolate their work. Branches allow you to work on different parts of a project without impacting the main branch.

**Create branches:** The main branch in git is called as master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the file which are created in branch are not shown in master branch. We can also merge both the parent (master) and child (other branches).

### Syntax:

For creating a new branch, git branch name by default is master branch.



### Snapshots:

```
navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git branch
* master

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ |
```

Default branch is master branch

```
navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git branch
* master

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git branch feature

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git branch
  feature
* master

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ |
```

### Adding a feature branch

```
navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git branch
* master

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git branch feature

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git branch
  feature
* master

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git checkout feature
Switched to branch 'feature'

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (feature)
$ git branch
  feature
* master

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (feature)
$ |
```

### Switching to feature branch

```

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git branch
  feature
* master

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git checkout feature
Switched to branch 'feature'

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (feature)
$ git branch
  feature
* master

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (feature)
$ git checkout master
Switched to branch 'master'

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git branch
  feature
* master

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$

```

### Switching to master branch

```

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (feature)
$ git checkout master
Switched to branch 'master'

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git branch
  feature
* master

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$ git log --oneline
6b418c6 (HEAD -> master, feature) This is my First Git Commit

navji@Navjinder MINGW64 ~/OneDrive/Desktop/navi singh (master)
$

```

### Checking commits via log

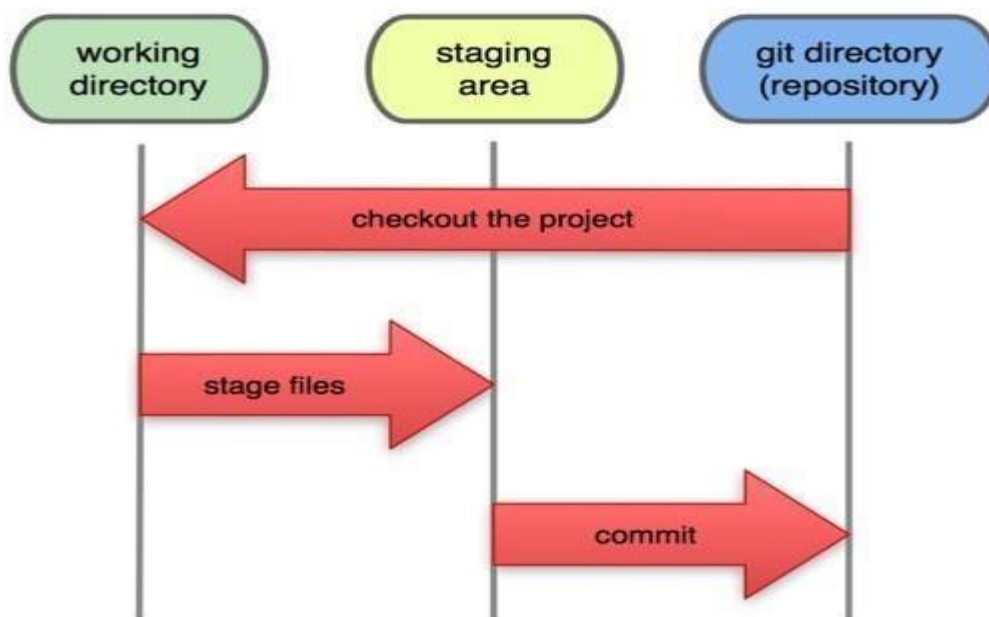
# EXPERIMENT - 5

**Aim:** Git lifecycle description

## Theory:

Stages in GIT Life Cycle: Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, when a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states:

- Working directory
- Staging area
- Git directory



## Working Directory:

Consider a project residing in your local system. This project may or may not be tracked by Git. To track the files in the directory we need to initialize the repository by using '*git init*' command. In either case, this project directory is called your Working directory.

## Staging Area:

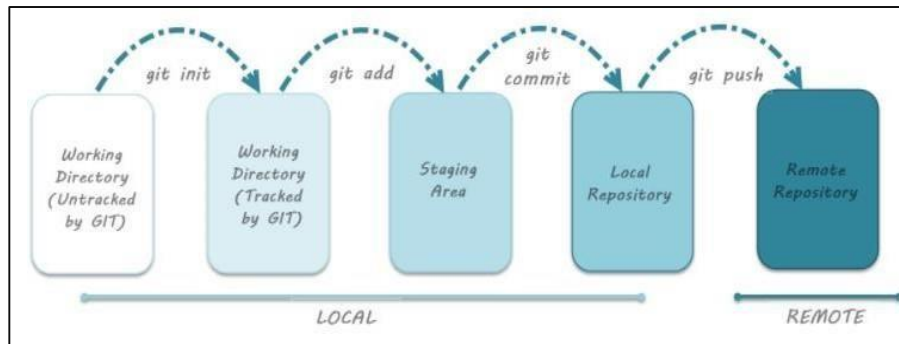
Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions. Files are added into the staging area by using '*Git add "filename"*' command.

## Git Directory:

Now that the files to be committed are grouped and ready in the staging area, we can commit these files. So, we commit this group of files along with a commit message explaining what is the commit about. Apart from commit message, this step also records the author and time of the commit. Now, a snapshot of the files in the commit is recorded by Git. The information related to this commit is stored in the Git directory. Files are added into directory in unmodified state by using '`git commit`' command, this will open the VIM editor where user can enter the commit message by using a few key combinations mainly '`i`' enter the message and then '`:wq`' to exit the VIM editor.

## Remote Repository:

It means mirror or clone of the local Git repository in GitHub. And pushing means uploading the commits from local Git repository to remote repository hosted in GitHub.



## File Status Lifecycle

