# SOURCE CODE MANAGEMENT FILE

**Submitted by:**
**Name:** Aakansha Pundir
**Roll No.:** 2310992577
**Group:** 22-B

**SubmittedTo:**
Dr. Sharad Chauhan ,
Professor ,CSE,
Chitkara University,
Punjab

**Submission of:** Task1.1
**Subject Name:** Source Code Management
**Sub Code:** CS181
**Department:** DCSE

# LIST OF PROGRAMS



CHITKARA UNIVERSITY
PUNJAB

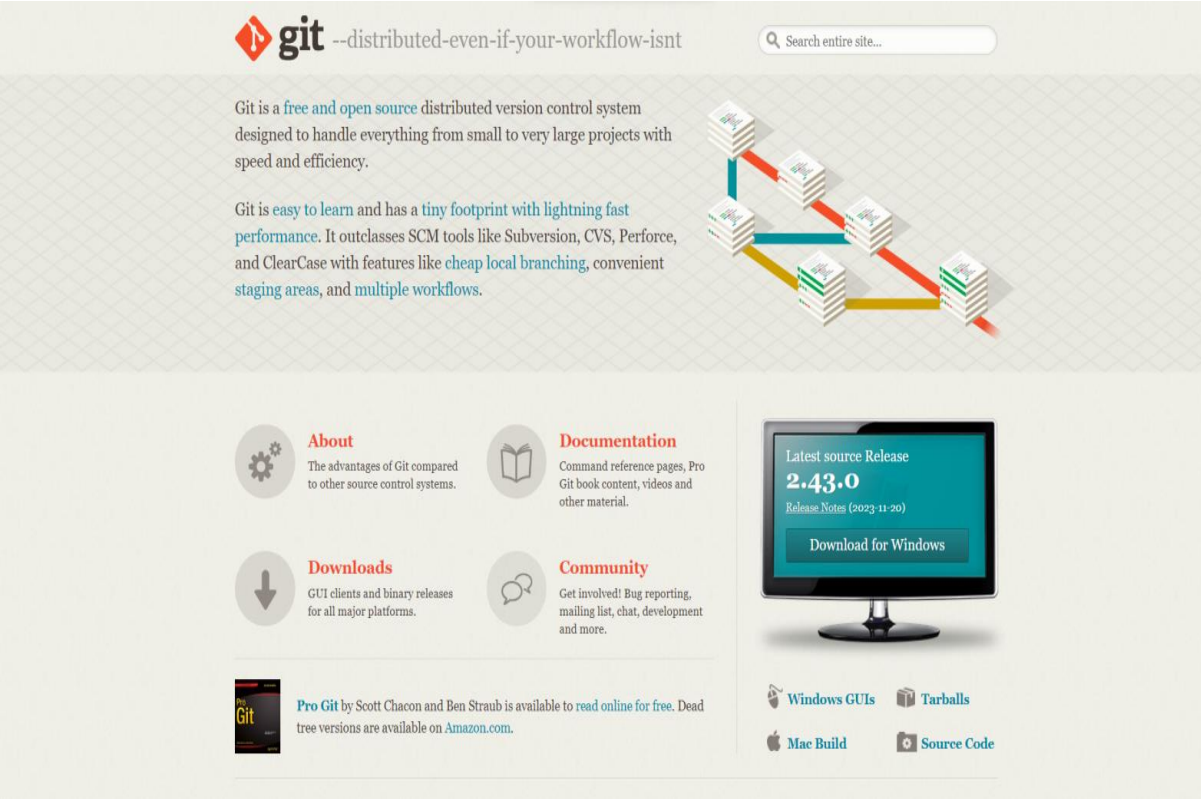
# Experiment No. 01

**Aim:** Setting up of Git Client

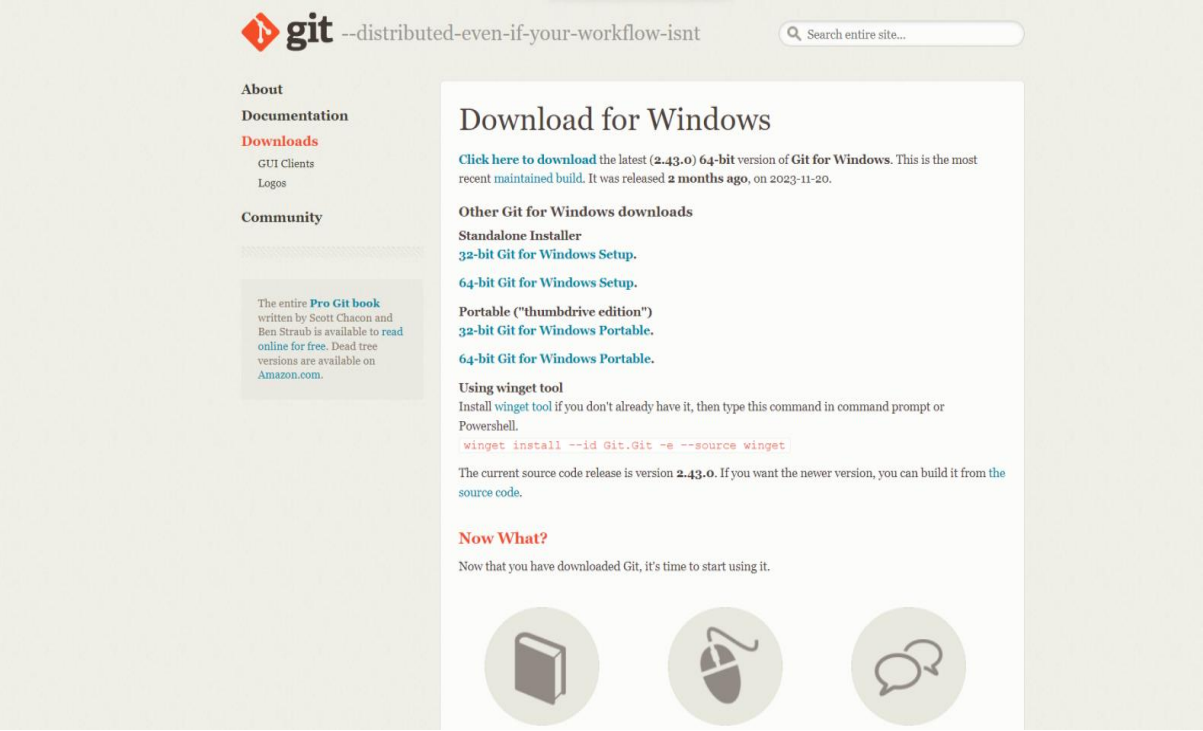
**Theory:**
GIT: It's a Version Control System (VCS). It is a software or we can say a server by which we are able to track all the previous changes in the code. It is basically used for pushing and pulling of code. We can use git and git-hub parallelly to work with multiple members or individually.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

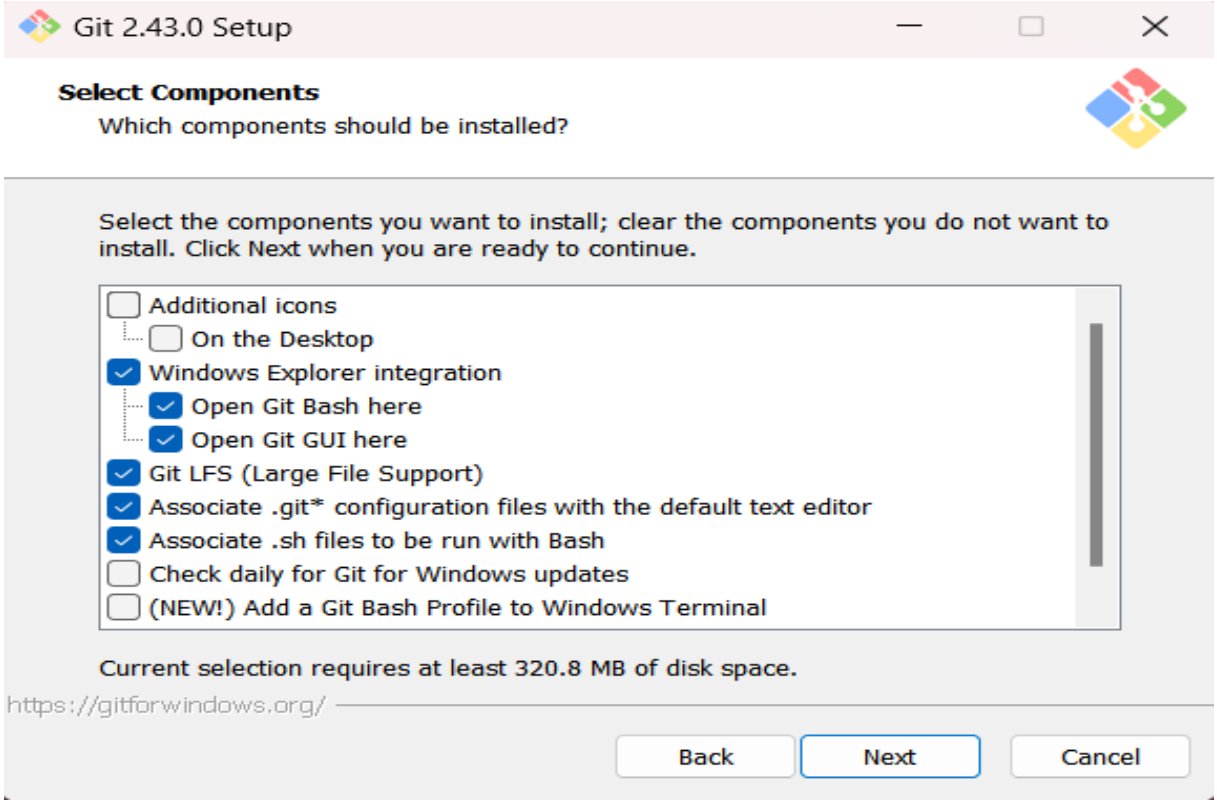
**Snapshots of download:**



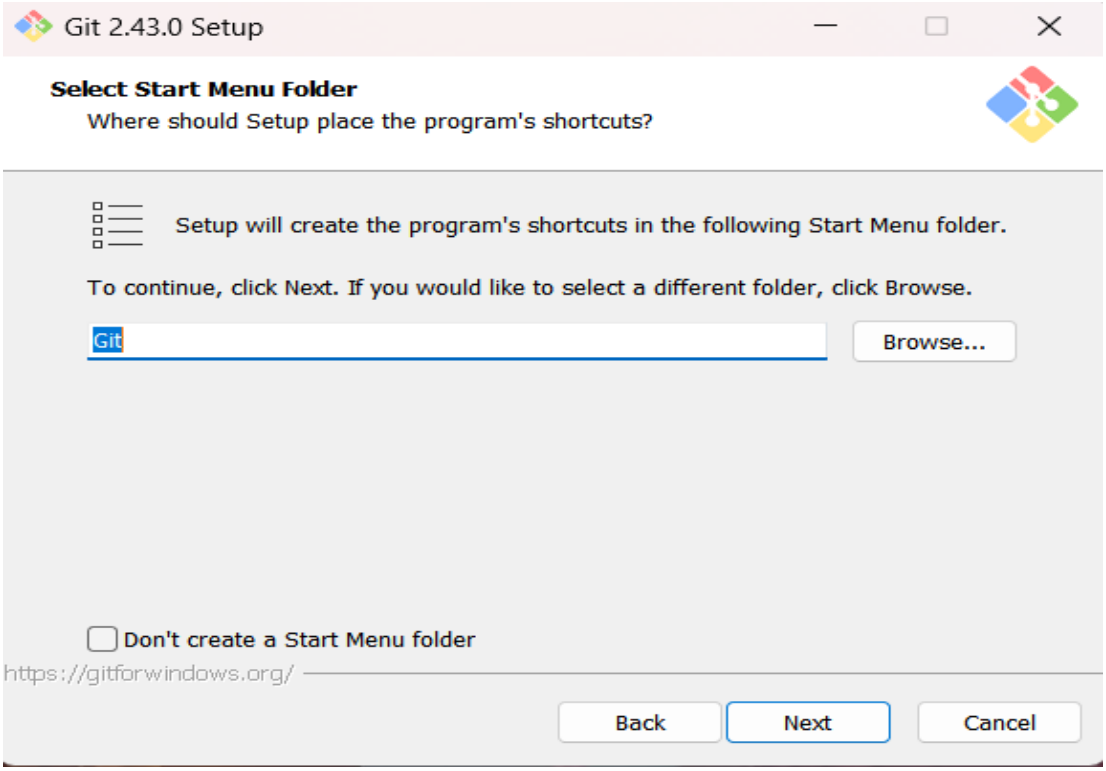

Installation of GIT



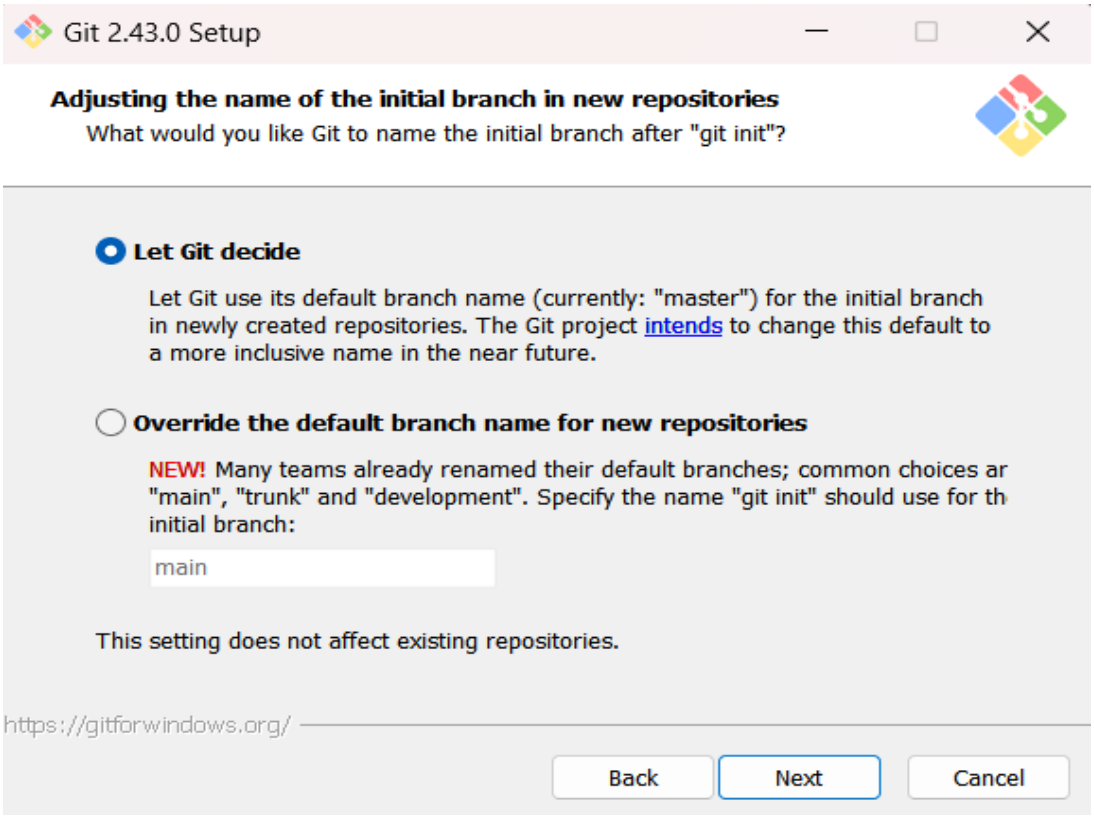

Download for Windows



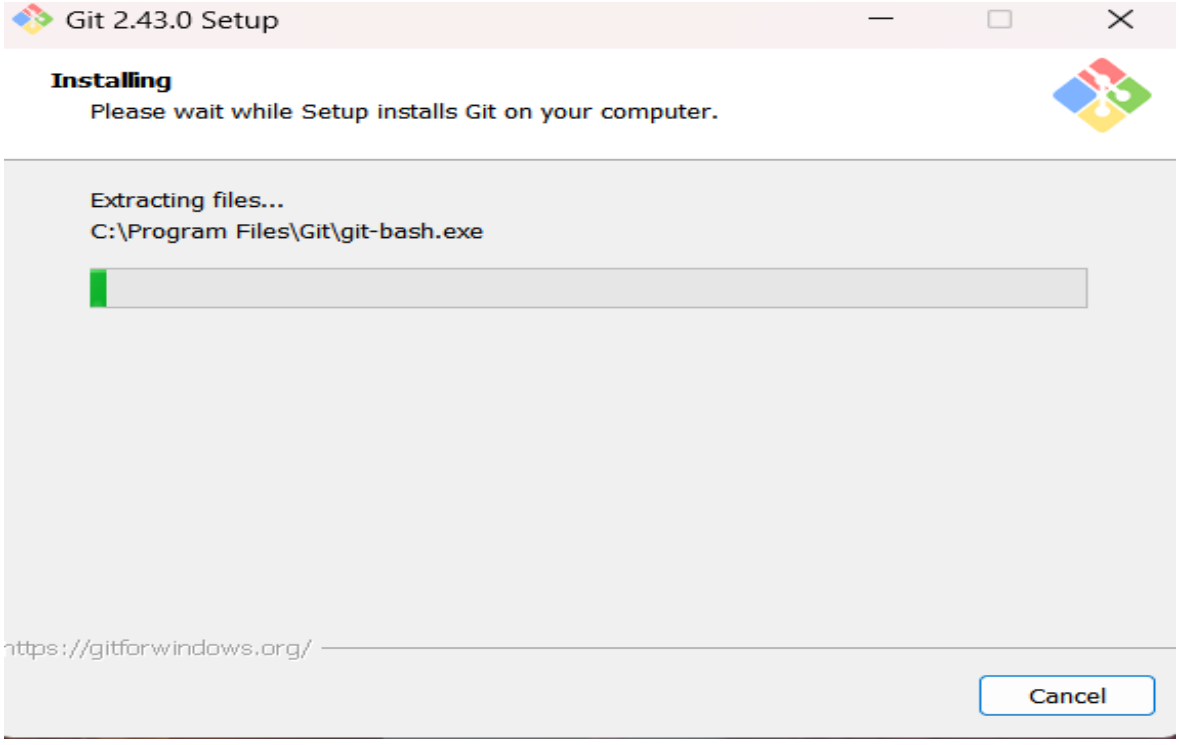

Select Components



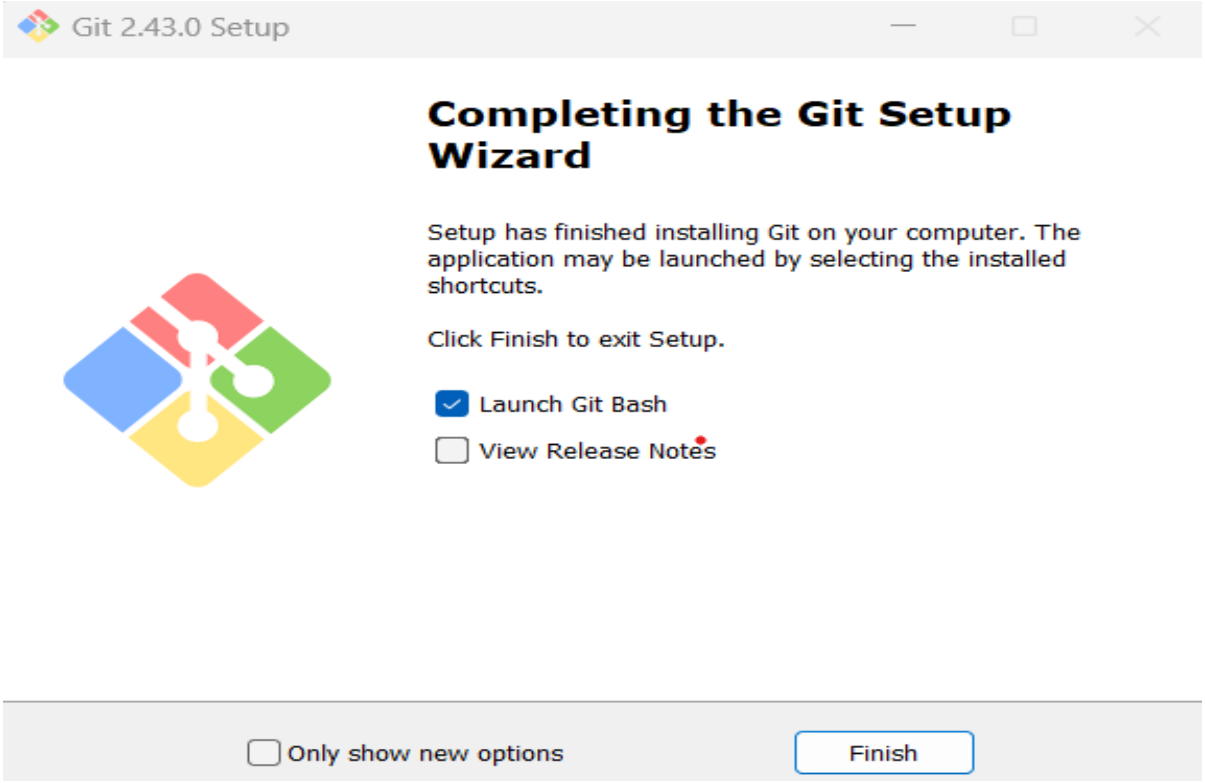

Select Star Menu Folder




Adjusting the name of the initial branch in new repositories




Installing




Completing the Git Setup Wizard

# Experiment No. 02

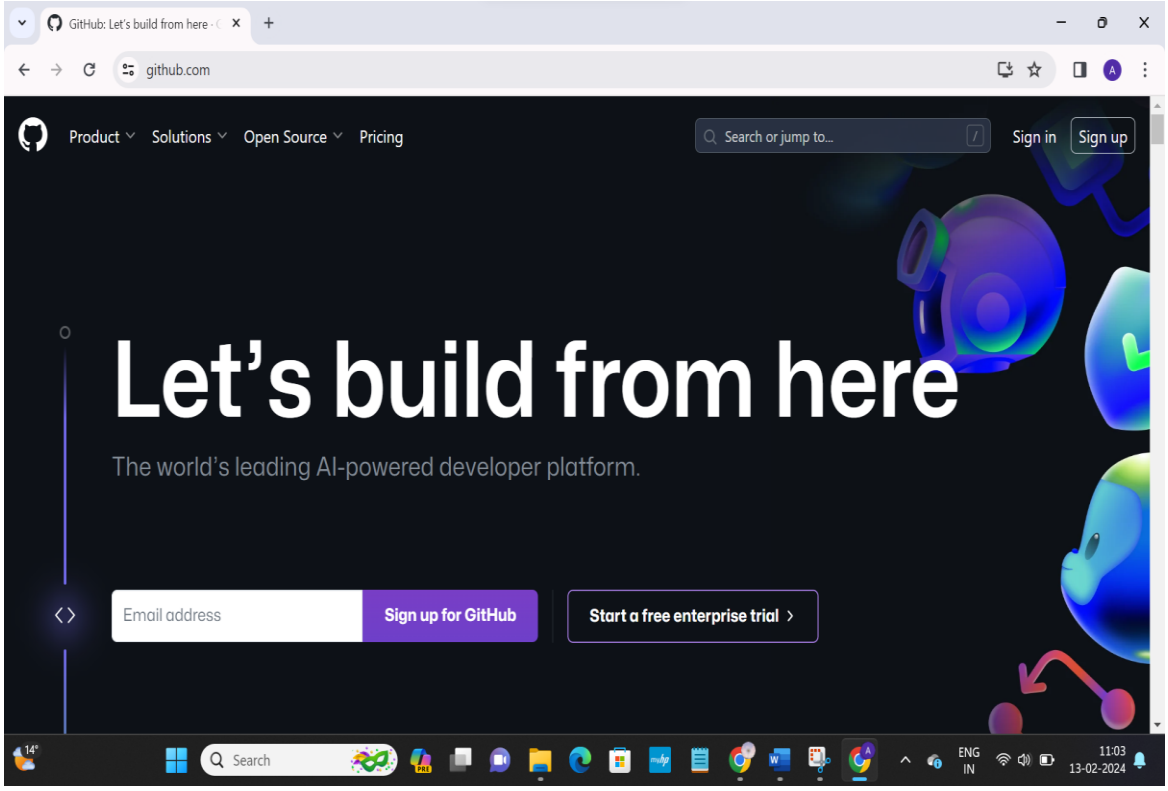
**Aim:** Setting up GitHub Account

**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.

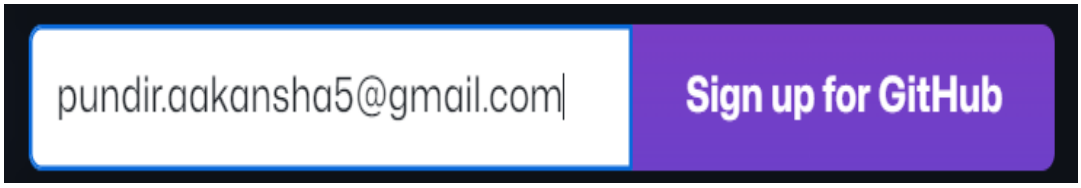
Advantages of GitHub: GitHub has a user-friendly interface and is easy to use. We can connect the git-hub and git but using some commands shown below in figure 001. Without GitHub we cannot use Git because it generally requires a host and if we are working for a project, we need to share it will our 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.

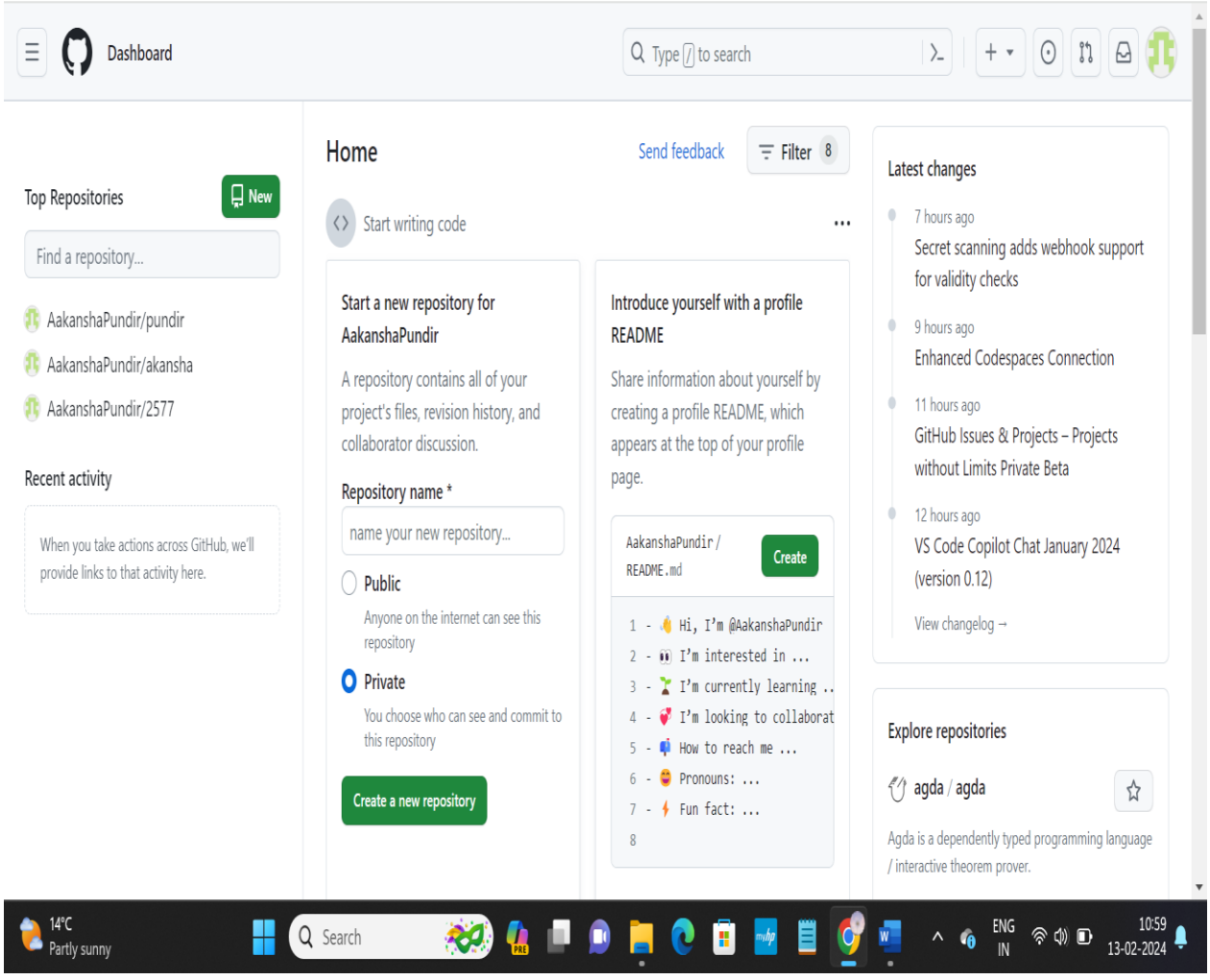
**Snapshots :**




Welcome to GitHub




Sign up for GitHub




GitHub Interface

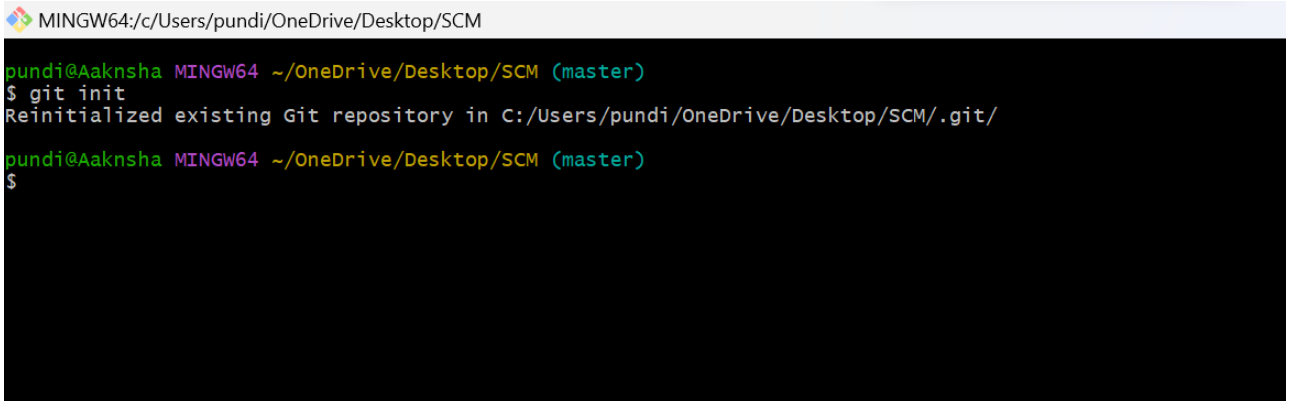
# Experiment No. 03

**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".




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:

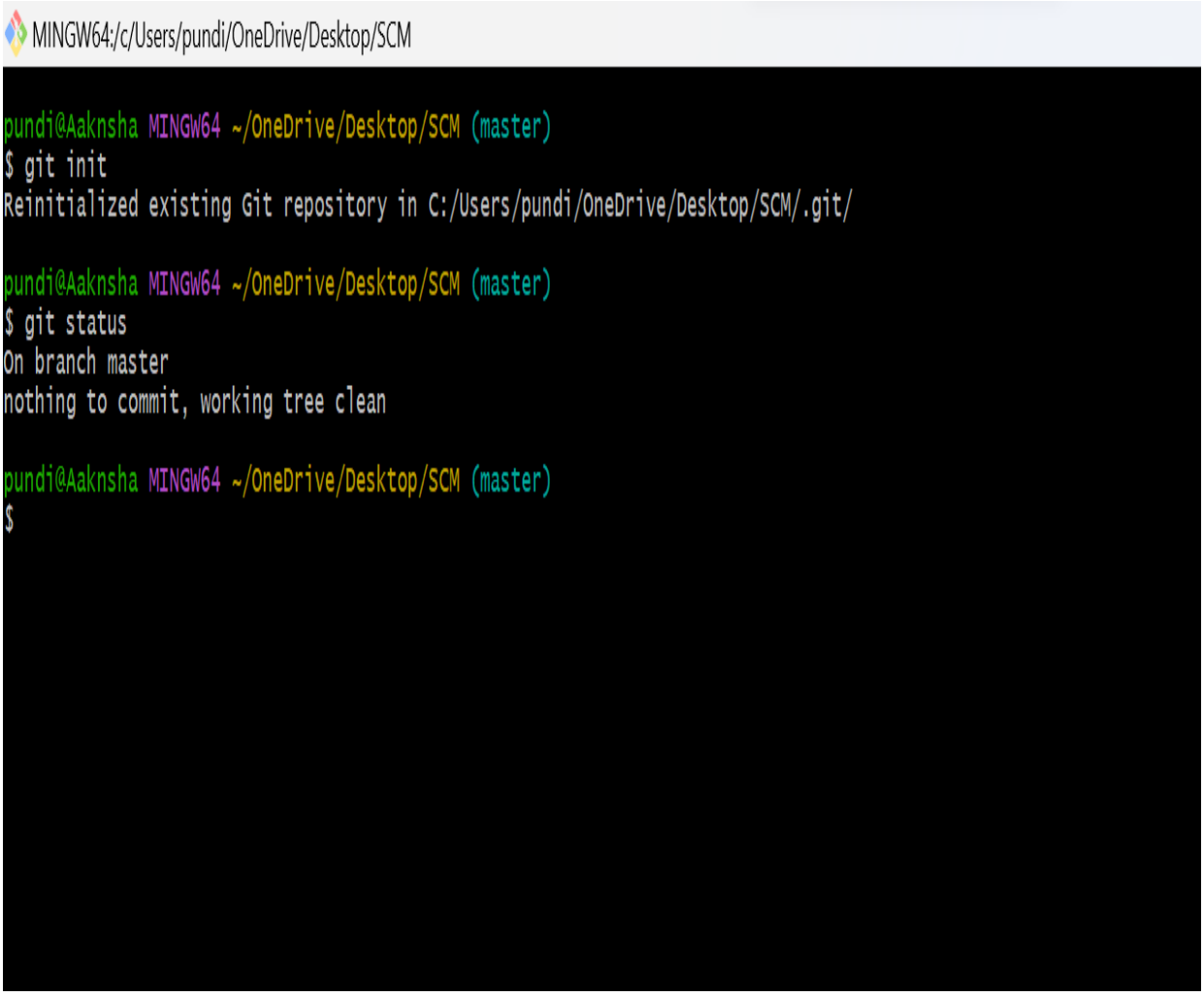
"git config --global user.name Name"
"git config --global user.email email"

For verifying the user's name and email, we use:

"git config --global user.name"
"git config --global user.email"

# SOME IMPORTANT COMMANDS

- Is :- It gives the file names in the folder.
- Is -lart :- Gives the hidden files also.
- Git status :- Displays the status 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

```
MINGW64:/c/Users/pundi/OneDrive/Desktop/SCM
$ git init
Reinitialized existing Git repository in C:/Users/pundi/OneDrive/Desktop/SCM/.git/

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git status
On branch master
nothing to commit, working tree clean

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git log
commit 18cdb2de66dd37d5f4b20fac71ef90c1bab450b1 (HEAD -> master)
Author: Aakansha Pundir <pundir.aakansha5@gmail.com>
Date:   Thu Feb 1 13:48:19 2024 +0530

    commit4:Fourth

commit 9c5387df65a60277e881b6cb479d920183195216
Author: Aakansha Pundir <pundir.aakansha5@gmail.com>
Date:   Thu Feb 1 13:47:11 2024 +0530

    commit-3:Third

commit b2925c31c5c0f88f133a6f3a63e23d89f1555ed3
Author: Aakansha Pundir <pundir.aakansha5@gmail.com>
Date:   Thu Feb 1 13:45:24 2024 +0530

    commit-2:Second

commit c338dbcc5c73bda65f44ece9bceb81cfe54c56bd
Author: Aakansha Pundir <pundir.aakansha5@gmail.com>
Date:   Thu Feb 1 13:38:11 2024 +0530

    commit-1:Initial

commit 7a4f8f295835f0234feedba54785cdac3dbb97aa
Author: Aakansha Pundir <pundir.aakansha5@gmail.com>
Date:   Wed Jan 31 11:22:00 2024 +0530

    commit-2:Second phase

commit df246dc8eec4535a8d21f14cee133fdadf231577
Author: Aakansha Pundir <pundir.aakansha5@gmail.com>
Date:   Wed Jan 31 11:15:20 2024 +0530

    commit-1: Initial

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$
```

Git log

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 metadat

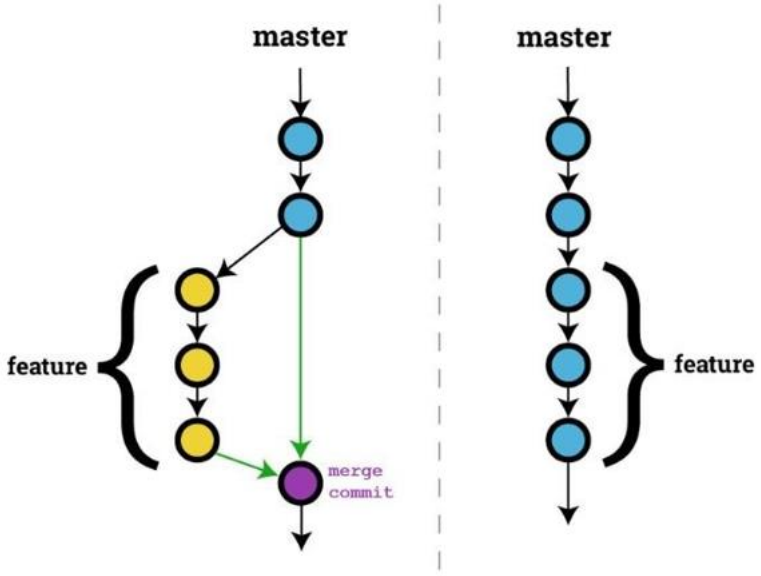
# Experiment No. 04

**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:

```
MINGW64:/c/Users/pundi/OneDrive/Desktop/SCM

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$
```

Default branch is master branch

```
MINGW64:/c/Users/pundi/OneDrive/Desktop/SCM

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch feature

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
  feature
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$
```

Adding a feature branch

```
MINGW64:/c/Users/pundi/OneDrive/Desktop/SCM

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch feature

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
  feature
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git checkout feature
Switched to branch 'feature'

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (feature)
$ git branch
* feature
  master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (feature)
$ |
```

Switching to feature branch

MINGW64:/c/Users/pundi/OneDrive/Desktop/SCM

```
pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch feature

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git branch
  feature
* master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git checkout feature
Switched to branch 'feature'

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (feature)
$ git branch
* feature
  master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (feature)
$ git checkout master
Switched to branch 'master'

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ |
```

Switching to master branch

```
pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (feature)
$ git branch
* feature
  master

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (feature)
$ git checkout master
Switched to branch 'master'

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$ git log --oneline
18cdb2d (HEAD -> master, feature) commit4:Fourth
9c5387d commit-3:Third
b2925c3 commit-2:Second
c338dbc commit-1:Initial
7a4f8f2 commit-2:Second phase
df246dc commit-1: Initial

pundi@Aaknsha MINGW64 ~/OneDrive/Desktop/SCM (master)
$
```

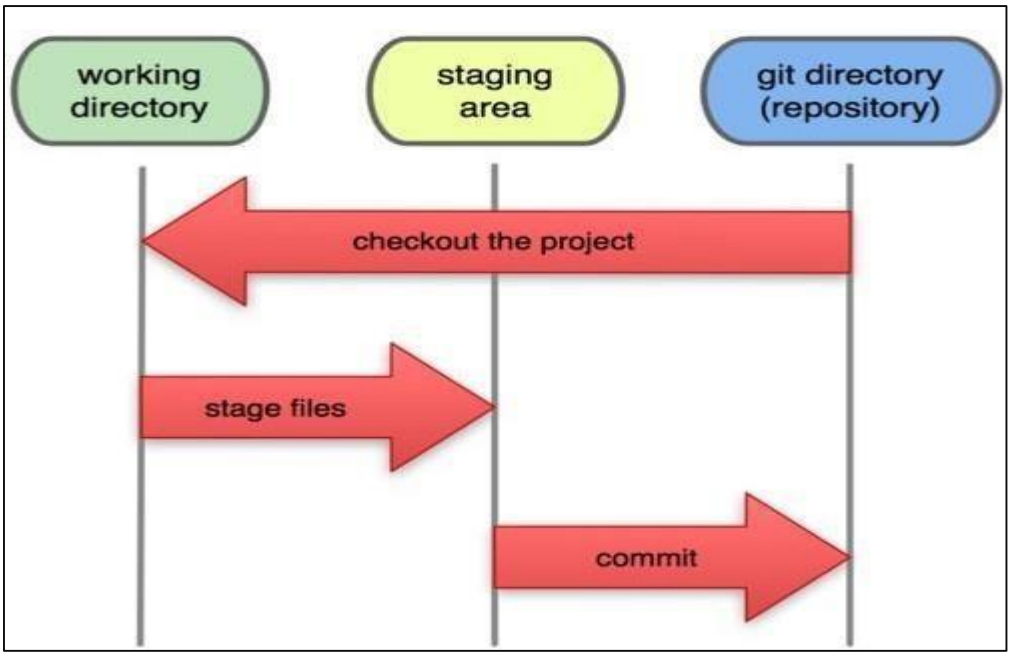
Checking commits via log

# Experiment No. 05

**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.