Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **DCSE**

Submitted By:
Akshay Jha
2110990127
G2-B

Submitted To:
Dr. Deepak Thakur

What is GIT and why is it used?

Git is a source code management technology used by DevOps. Git is a piece of software that allows you to track changes in any group of files. It is a free and open-source version control system that may be used to efficiently manage small to big projects. Git is a version control system that allows numerous developers to collaborate on non-linear development projects.
Git is an example of a distributed version control system (DVCS) (hence Distributed Version Control System).

What is GITHUB?

GitHub is a version management and collaboration tool for programming. It allows you and others to collaborate on projects from any location.

What is the difference between GIT and GITHUB?

Git	GitHub
1. It is a software	1. It is a service
2. It is installed locally on the system	2. It is hosted on Web
3. It is a command line tool	3. It provides a graphical interface
4. It is a tool to manage different versions of edits, made to files in a git repository	4. It is a space to upload a copy of the Git repository
5. It provides functionalities like Version Control System Source Code Management	5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features

What is Repository?

A repository stores all of your project's files, as well as the revision history for each one. Within the repository, you may discuss and monitor your project's progress. The.git/ subdirectory within a project is a Git repository. This repository keeps track of any changes made to files in your project over time, creating a history. That is, if you delete the.git/ subdirectory, you are also deleting the history of your project.

What is Version Control System (VCS)?

Version Control Systems are the software tools for tracking/managing all the changes made to the source code during the project development. It keeps a record of every single change made to the code. It also allows us to turn back to the previous version of the code if any mistake is made in the current version. Without a VCS in place, it would not be possible to monitor the development of the project.

Types of VCS

- Local Version Control System

- Centralized Version Control System
- Distributed Version Control System

I. **Local Version Control System:** Local Version Control System is located in your local machine. If the local machine crashes, it would not be possible to retrieve the files, and all the information will be lost. If anything happens to a single version, all the versions made after that will be lost.

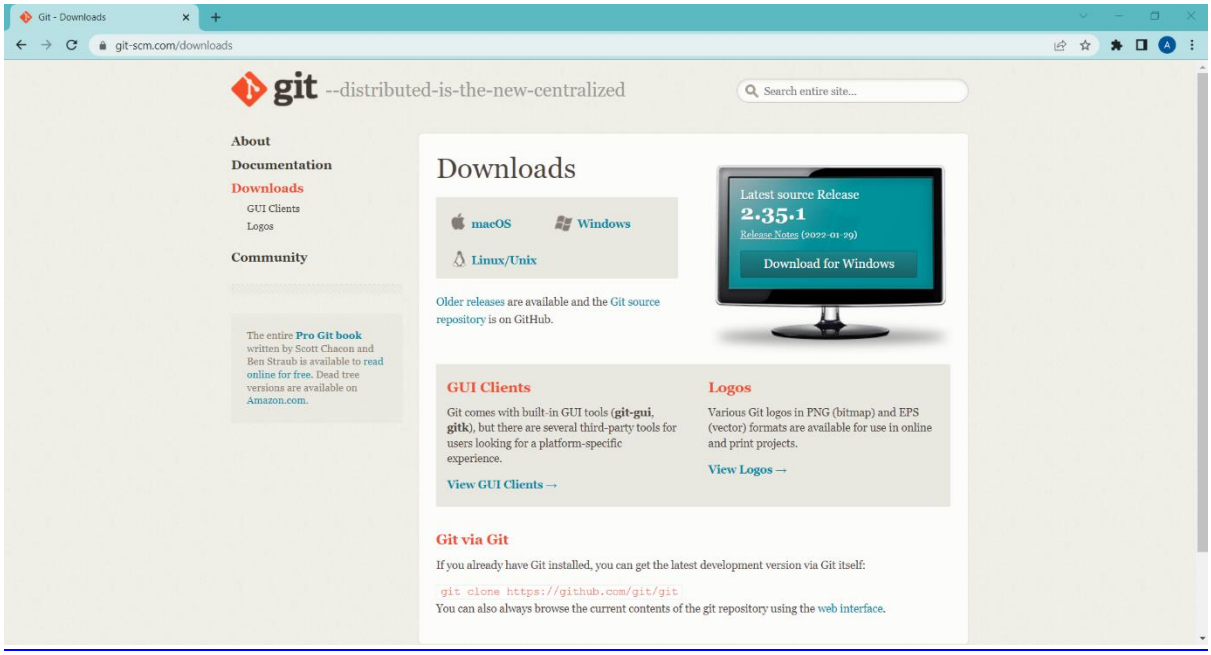
II. **Centralized Version Control System:** In the Centralized Version Control system, there will be a single central server that contains all the files related to the project, and many collaborators checkout files from this single server (you will only have a working copy). The problem with the Centralized Version Control Systems is if the central server crashes, almost everything related to the project will be lost.

III. **Distributed Version Control System:** In a distributed version control system, there will be one or more servers and many collaborators similar to the centralized system. But the difference is, not only do they check out the latest version, but each collaborator will have an exact copy of the main repository on their local machines. Each user has their own repository and a working copy. This is very useful because even if the server crashes we would not lose everything as several copies are residing in several other computers.

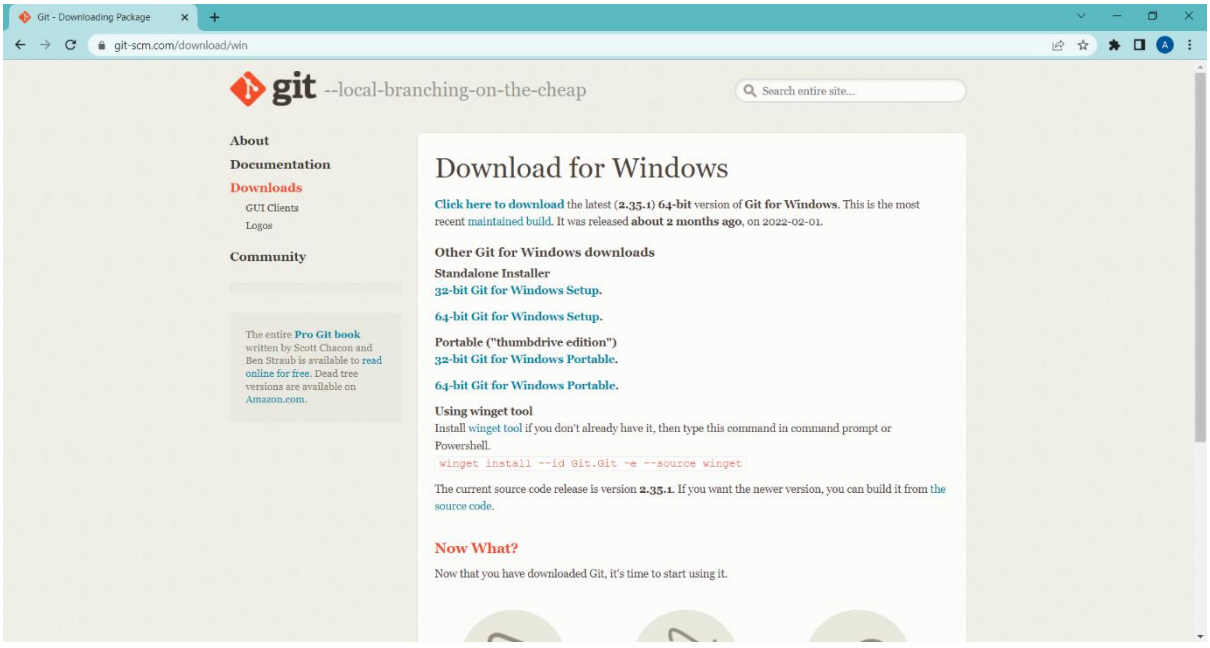
Experiment No. 01

Aim: Setting up of Git Client

✧ For git installation on your system, go to the linked URL.
https://git-scm.com/downloads



✧ You must first access this webpage and then choose your operating system by clicking on it. I'll walk you through the processes for the Windows operating system in this article.



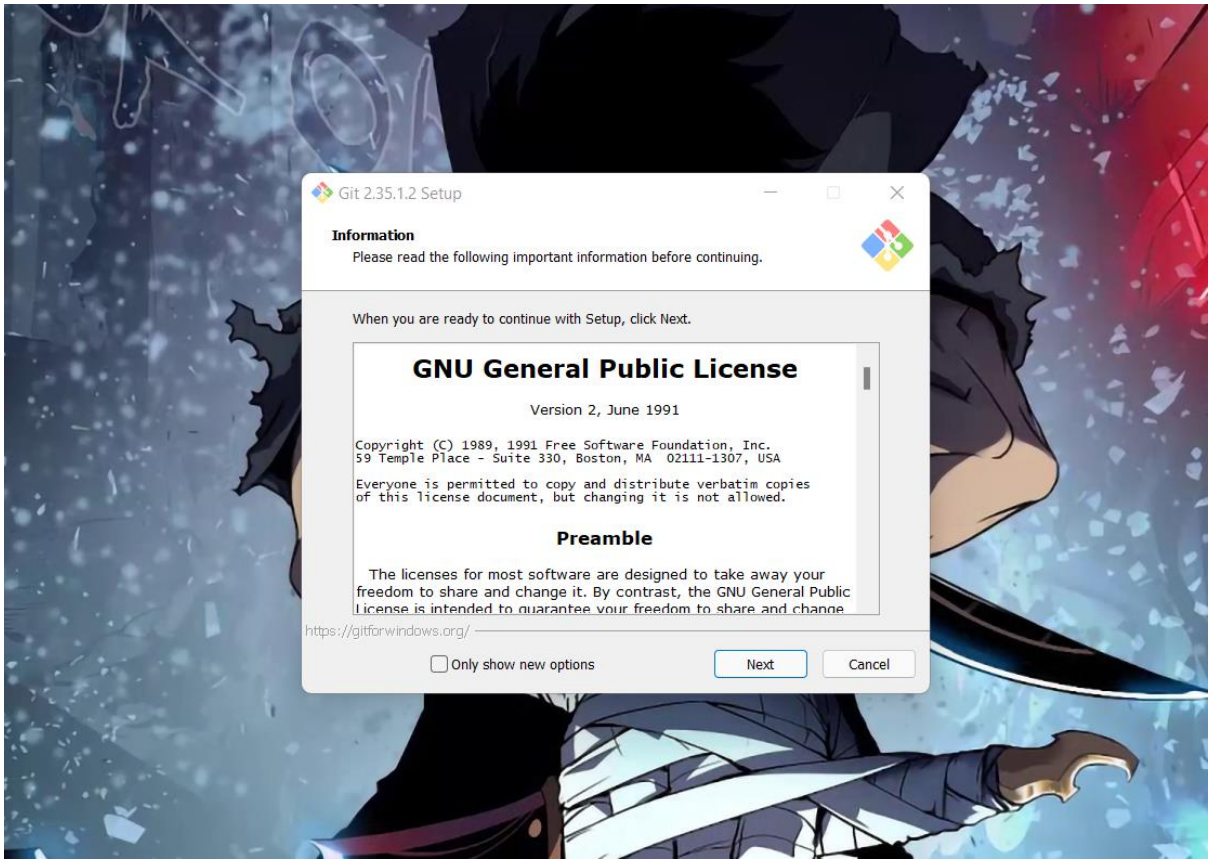
✧ Select the CPU for your system now. (Most of the system now runs on

64-bit processors.) Your download will begin when you pick a processor.

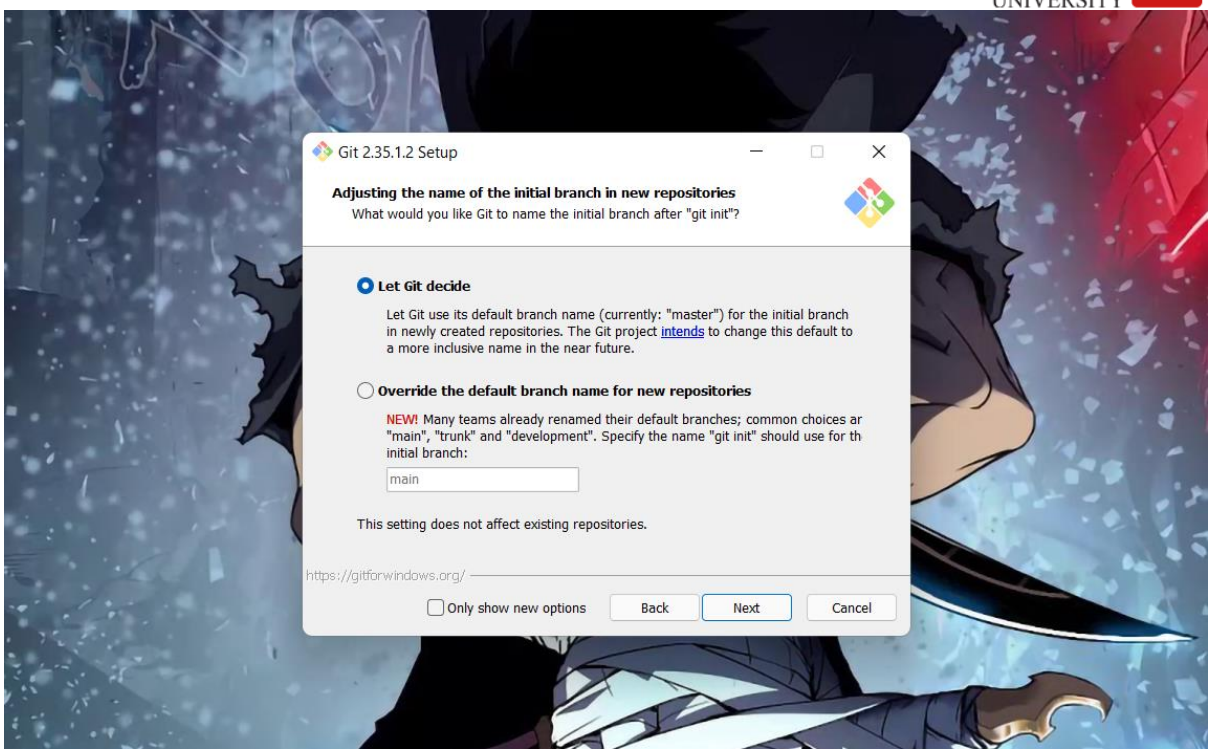
✧ You must now open the Git folder.

✧ You will be asked if you want to enable this program to make modifications to your PC once you launch it.

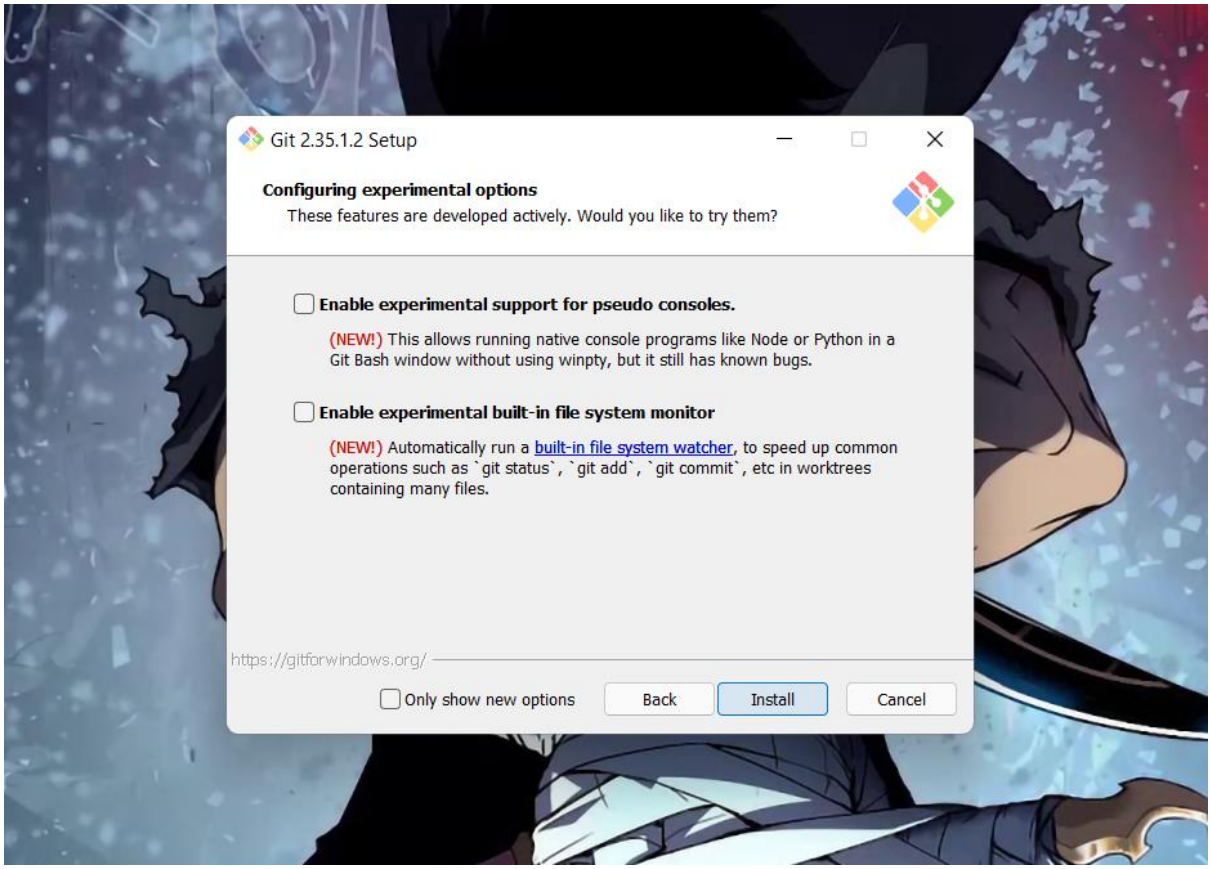
✧ YES should be selected.



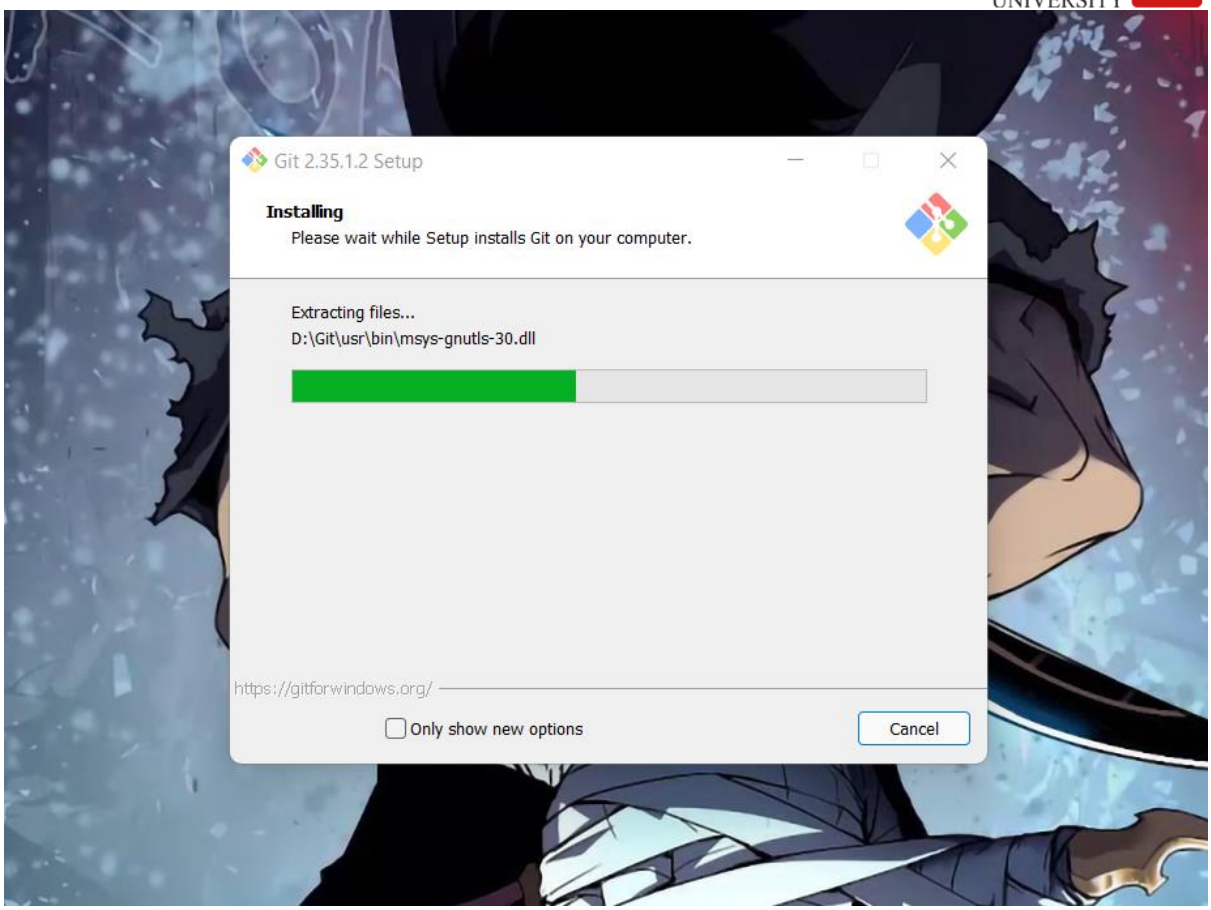
✧ Click on Next



✧ Continue clicking on next few times more



✧ Now select the Install option.



✧ Click on Finish after the installation is finished.

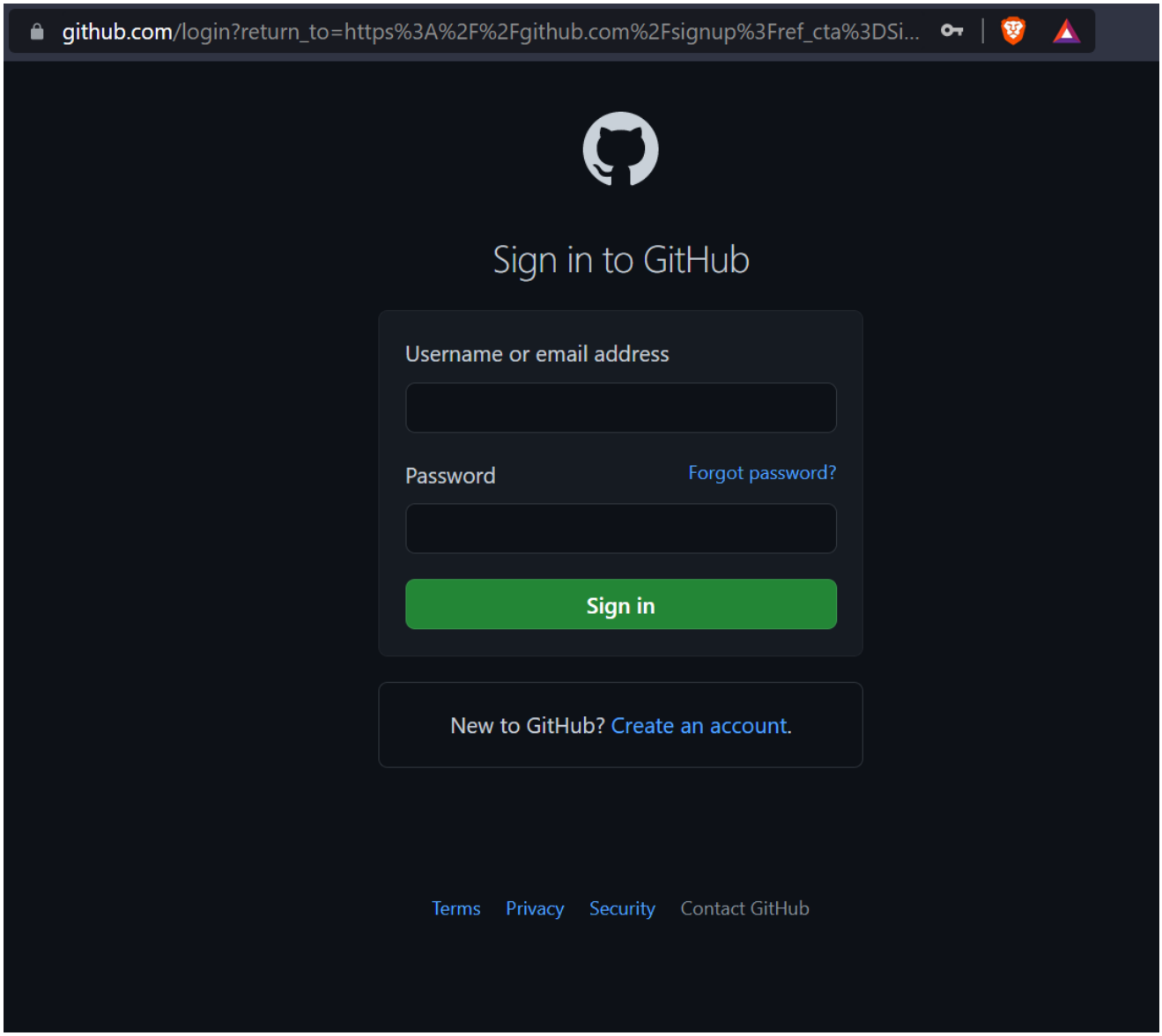
The installation of the git is finished and now we have to setup git client and GitHub account.

Experiment No. 02

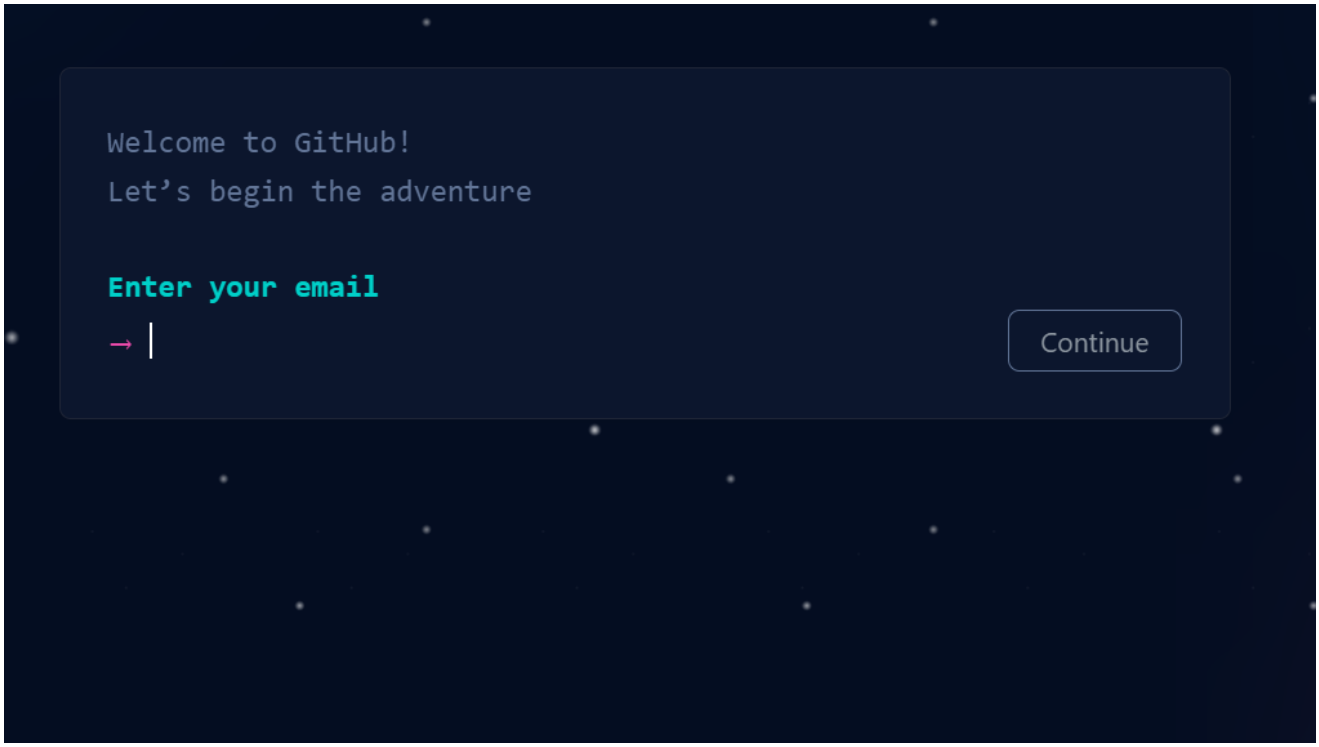
Aim: Setting up GitHub Account

✧ Open your web browser search GitHub login.

✧ Click on Create an account if you are a new user or if you have already an account, please log in.

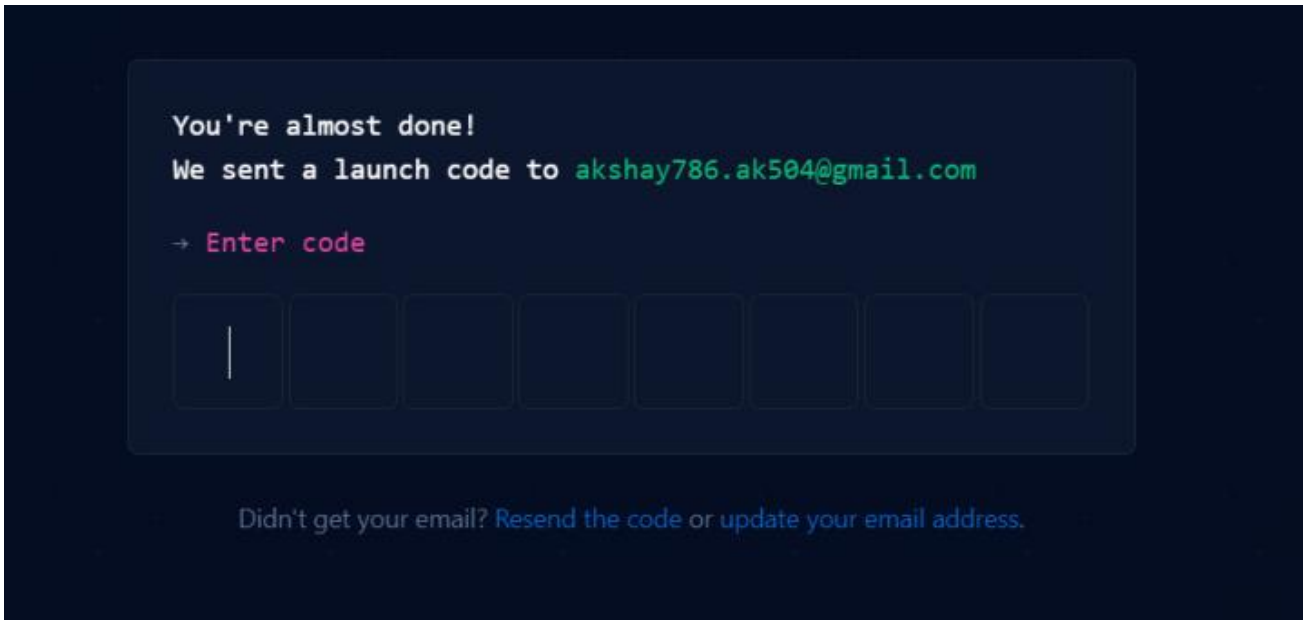


✧ After clicking on "Create a New Account," you will be sent to a new page where you must enter your email address for your account. Now type in the password you'd want to use for your GitHub account. Then you'll be prompted to enter your username.



✧ Now Click on Create Account.

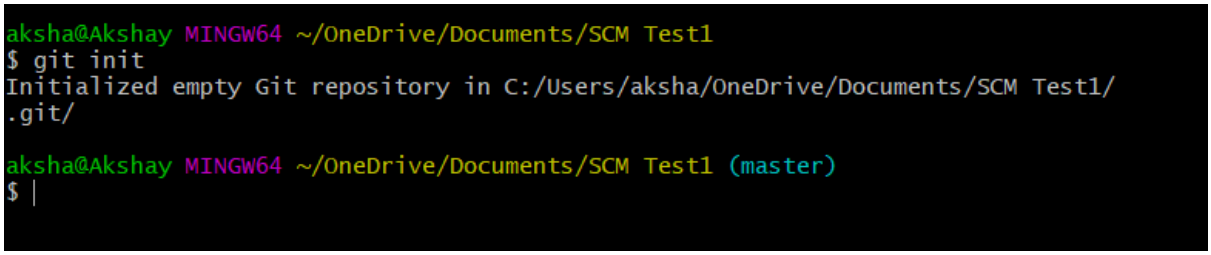
✧ Verify it from your email and you are all set to go.



Experiment No. 03

Aim: Program to Generate logs

✧ First of all, create a local repository using Git. For this, you have to make a folder on 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 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 the command →

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

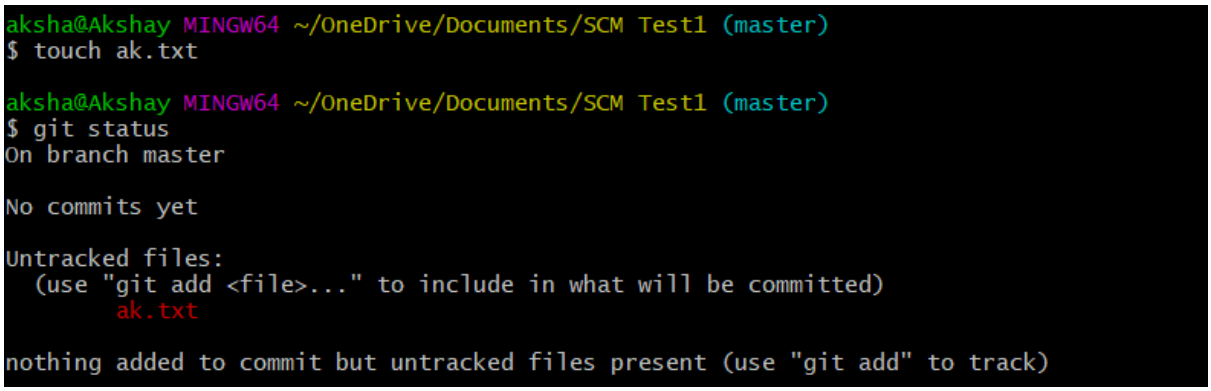
For verifying the user's name and email, we use →

"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 the staging area.

Now, we have to create some files in the repository. Suppose we created ak.txt Now type git status:



You can see that *ak. txt* is in red colour which means it is an untracked file. Now firstly add the file in the staging area and then commit the file.

For this, use command →

`git add -A` [For add all the files in staging area.]

`git commit -m "write any message"` [For commit the file]

```
aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$ git add -A

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$ git status
On branch master

No commits yet

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


aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$ git commit -m "Init commit"
[master (root-commit) 70e060b] Init commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ak.txt

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$ git status
On branch master
nothing to commit, working tree clean
```

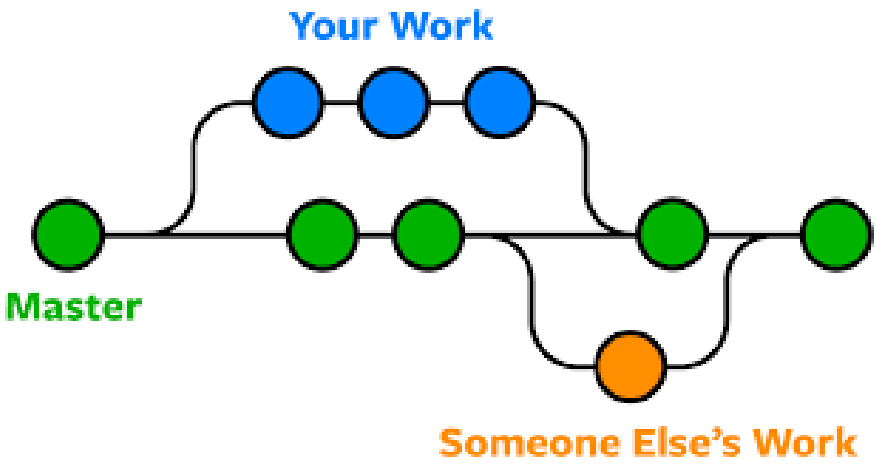
✧ **git log:** *The git log command displays a record of the commits in a Git repository. By default, the git log command displays a commit hash, the commit message, and other commit metadata.*

```
aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$ git log
commit 226bf9b98c4c38e211991bc68749df3b1e3d9f0f (HEAD -> master)
Author: Akshay <akshay0127.be21@chitkara.edu.in>
Date:   Wed Mar 30 19:43:12 2022 +0530

    added file

commit 410d0583cd67baa4e636874bb2075f2e5eb174ed
Author: Akshay <akshay0127.be21@chitkara.edu.in>
Date:   Wed Mar 30 19:42:01 2022 +0530

    initial commit

commit 70e060b4cd47d84e22c2f657a2a47b0a1fb20cd0
Author: Akshay <akshay0127.be21@chitkara.edu.in>
Date:   Wed Mar 30 19:36:32 2022 +0530

    Init commit

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$
```

Experiment No. 04

Aim: Create and visualize branches

✧ **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.



Let us see the command of it:
Firstly, add a new branch, let us suppose the branch name is activity1.
For this use command →

- **git branch name** **[**adding new branch**]**
- **git branch** **[**use to see the branch's names**]**
- **git checkout *branch name*** **[**use to switch to the given branch**]**

```
aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$ git branch
  activity1
  activity2
  activiy3
* master

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$ git branch newset

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$ git checkout newset
Switched to branch 'newset'

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (newset)
$ git branch
  activity1
  activity2
  activiy3
  master
* newset
```

In this, you can see that firstly 'git branch' shows only one branch in green color but when we add a new branch using 'git branch activity1', it shows 2 branches but the green color and star is on the master. So, we have to switch to act1 by using 'git checkout act1'. If we use 'git branch', now you can see that the green colour and star is on newset. It means you are in newset branch and all the data of the master branch is also on newset branch. Use "ls" to see the files.

Now add a new file in newset branch, do some changes in file and commit the file.

```
aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (newset)
$ touch nt.txt

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (newset)
$ git add .

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (newset)
$ git commit -m "new files added"
[newset 35ef7a7] new files added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 nt.txt

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (newset)
$ git status
On branch newset
nothing to commit, working tree clean

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (newset)
$ ls
ak.txt  fg.txt  fkg.txt  nt.txt
```

If we switched to the master branch, 'nt.txt' file is not there. But the file is in newset branch.

```
aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (newset)
$ ls
ak.txt  fg.txt  fkg.txt  nt.txt

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (newset)
$ git checkout master
Switched to branch 'master'

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$ ls
ak.txt  fg.txt  fkg.txt

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$
```

- To add these files in master branch, we have to do merging. For this firstly switch to master branch and then use command →

  git merge branchname [use to merge branch]

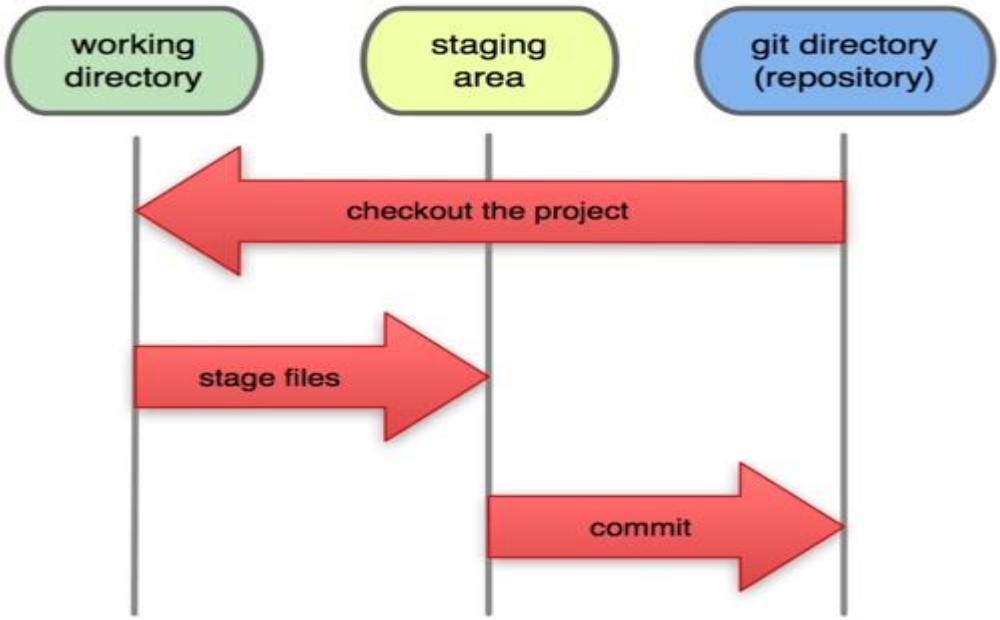
```
aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$ git merge newset
Updating 226bf9b..35ef7a7
Fast-forward
 nt.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 nt.txt

aksha@Akshay MINGW64 ~/OneDrive/Documents/SCM Test1 (master)
$ git log
commit 35ef7a7bc819dc3181294a429effee4368040938 (HEAD -> master, newset)
Author: Akshay <akshay0127.be21@chitkara.edu.in>
Date:   Wed Mar 30 19:56:16 2022 +0530

    new files added

commit 226bf9b98c4c38e211991bc68749df3b1e3d9f0f (activiy3, activity2, activity1)
Author: Akshay <akshay0127.be21@chitkara.edu.in>
Date:   Wed Mar 30 19:43:12 2022 +0530

    added file

commit 410d0583cd67baa4e636874bb2075f2e5eb174ed
Author: Akshay <akshay0127.be21@chitkara.edu.in>
Date:   Wed Mar 30 19:42:01 2022 +0530

    initial commit

commit 70e060b4cd47d84e22c2f657a2a47b0a1fb20cd0
Author: Akshay <akshay0127.be21@chitkara.edu.in>
Date:   Wed Mar 30 19:36:32 2022 +0530

    Init commit
```

Experiment No. 05

Aim: Git lifecycle description

Stages in GIT Life Cycle:

Now let's understand the three-stage architecture of Git:



- **Working Directory:** This is the directory that we've initialized, and here all the changes are made to commit on GitHub.

- **Staging Area:** This is where we first put out code or files of the working repository. The command that we use to stage code is, "git add --a", "git add FileName" or "git add -A". In simple terms, staging means telling Git what files we want to commit (new untracked files, modified files, or deleted files).

- **Git directory(repository):** This is where all the commits are stored whenever we make a commit. We can revert to an older version of or project using the "git checkout" command from this directory.

CHITKARA UNIVERSITY

INDEX