Subject Name: **Source Code Management**

Subject Code: **22CS003**

Session: **2023-24**

Department: **DCSE**

Submitted By:

Hunar bhutani

2310990440-First Year

G06-A

Submitted To:

Dr. Chetna Kaushal

List of Programs

EXPERIMENT:-1

Aim: Setting up of Git Client

Theory:

What is Git?

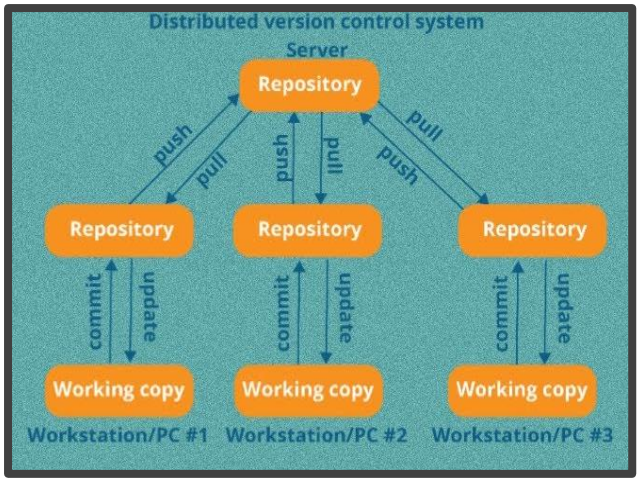
Git is a free and open-source version control system used to handle small to very large projects efficiently. This is also used for tracking changes in any set of files and usually helps in coordinating work among members of a team. Hence, enables multiple developers to work together on non-linear development.

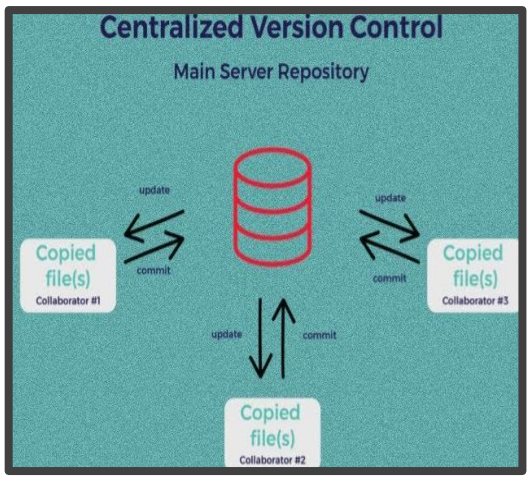
History of VCS: The very first Version Control System was created in 1972 at Bell Labs where they also developed UNIX. The first one was called SCCS (Source Code Control System). It was available only for UNIX and only worked with Source Code files. Some types of Version Control Systems are:

• Local VCS: No internet is needed because it uses a database to keep and track of files.

• Centralized VCS: Centralized version control systems are based on the idea that there is a single "central" copy of your project somewhere (probably on a server), and programmers will "commit" their changes to this central copy. This simply means recording the change in the central system (OS).

• Distributed VCS: A type of version control where the complete codebase including its full version history is mirrored on every developer's computer.

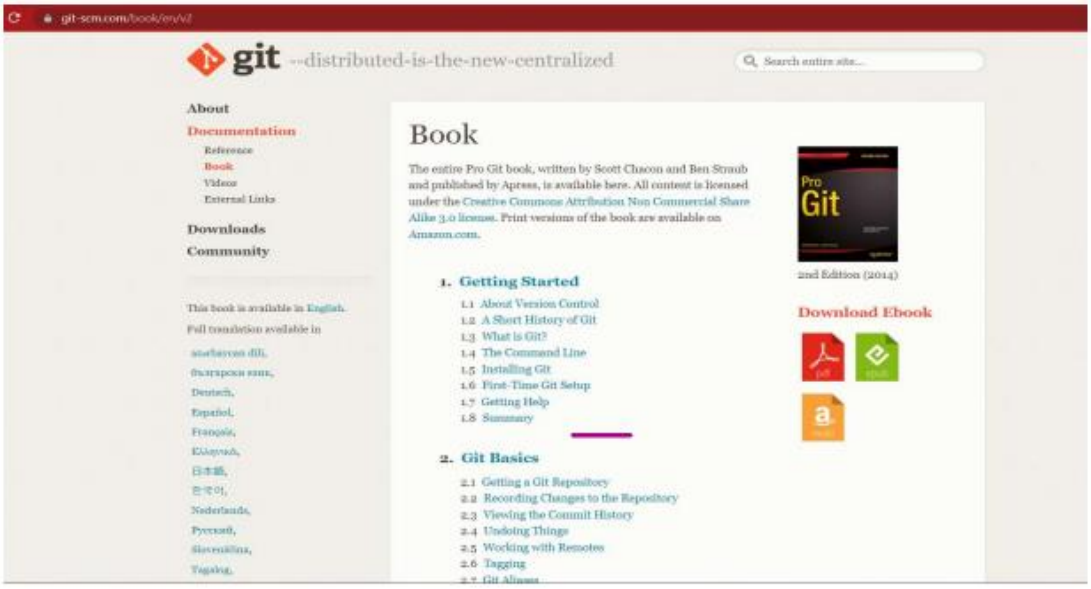




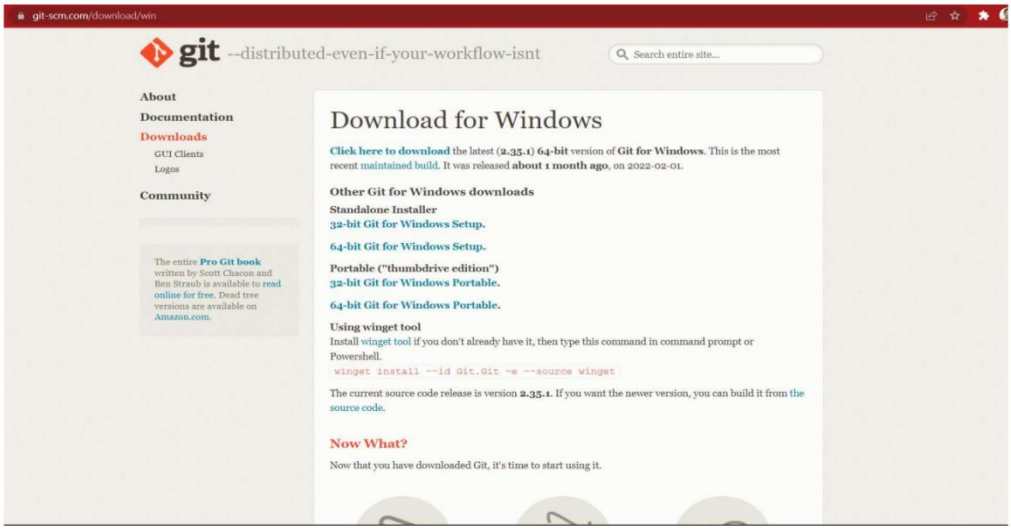
How to install GIT on Windows?

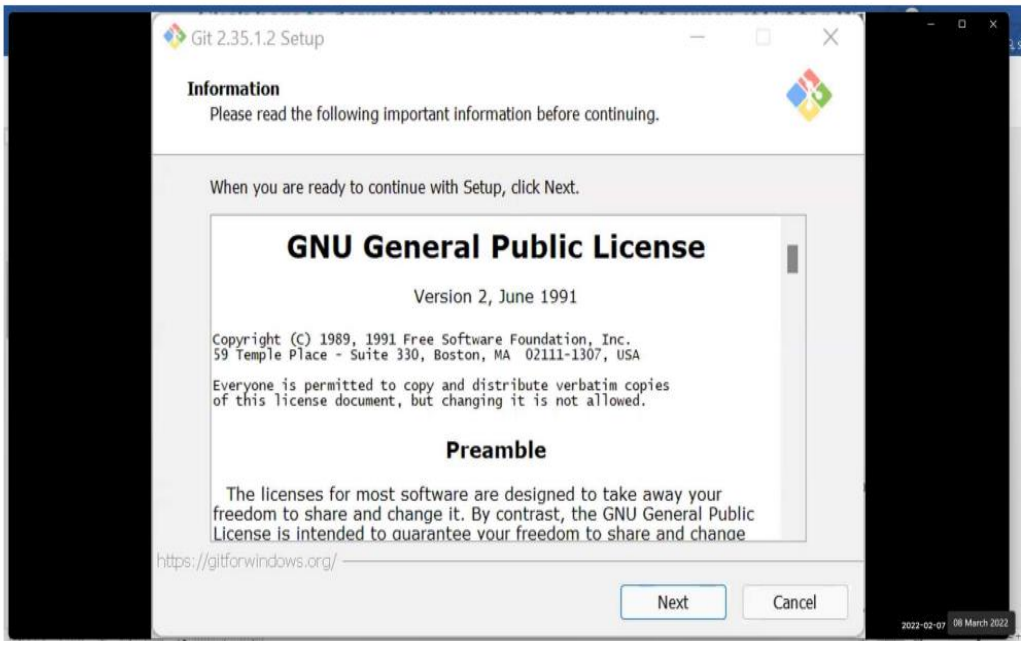
There are many ways to install Git on Windows. The most official build is available for download on the Git website. Go to https://gitscm.com/download/win and after a few settings the download will start automatically.

- Visit directly on git book page by https://git-scm.com/book/en/v2

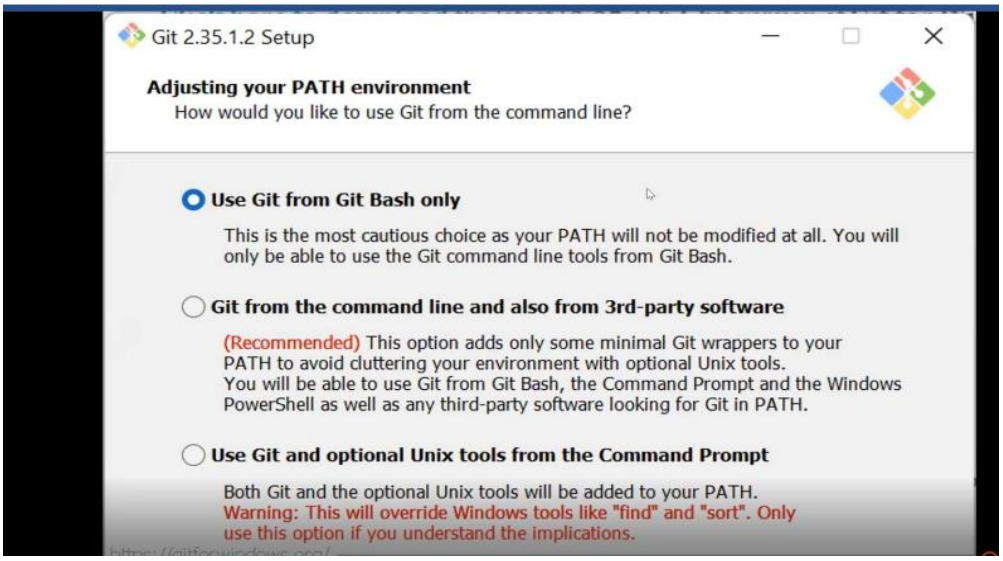


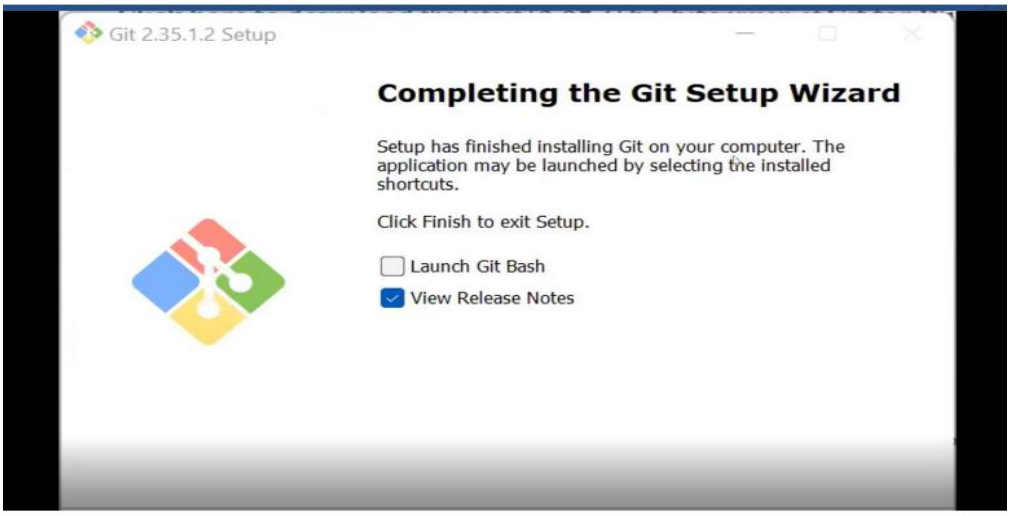
- Then click on Installation Git and click on whatever system you want, available are three- Windows, Apple and Linux.



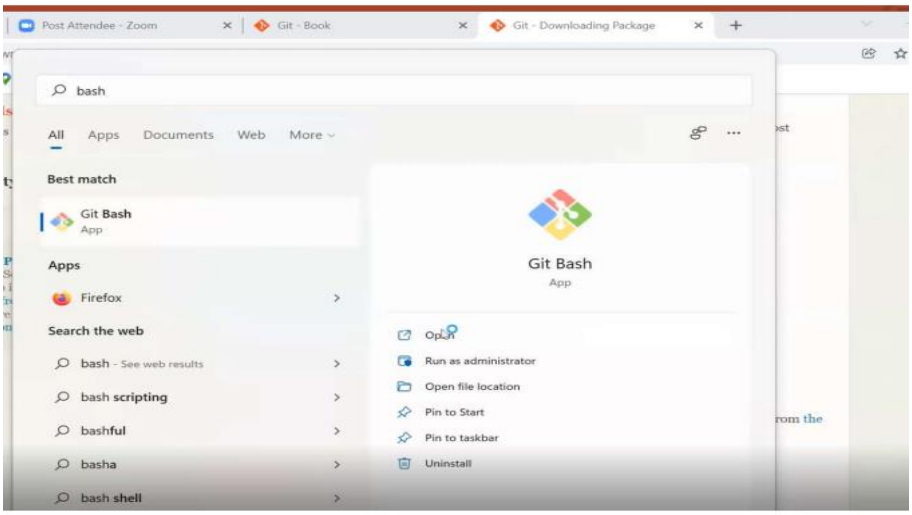


- After some more simple and easy settings and choosing your favourable environment and doing some SSH settings, it finally starts exporting the files in system and completes the Git hub wizard.





- Git bash got installed in system and seemed and opened on clicking seems of like:



```
Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git --version
git version 2.43.0.windows.1

Hp@HunarBhutani MINGW64 /e/scmFile (mast
er)
$
```

You can also check the version of installed software by checking git version.

EXPERIMENT NO 2:-

Aim: Setting up GitHub Account

Theory:

What is GitHub?

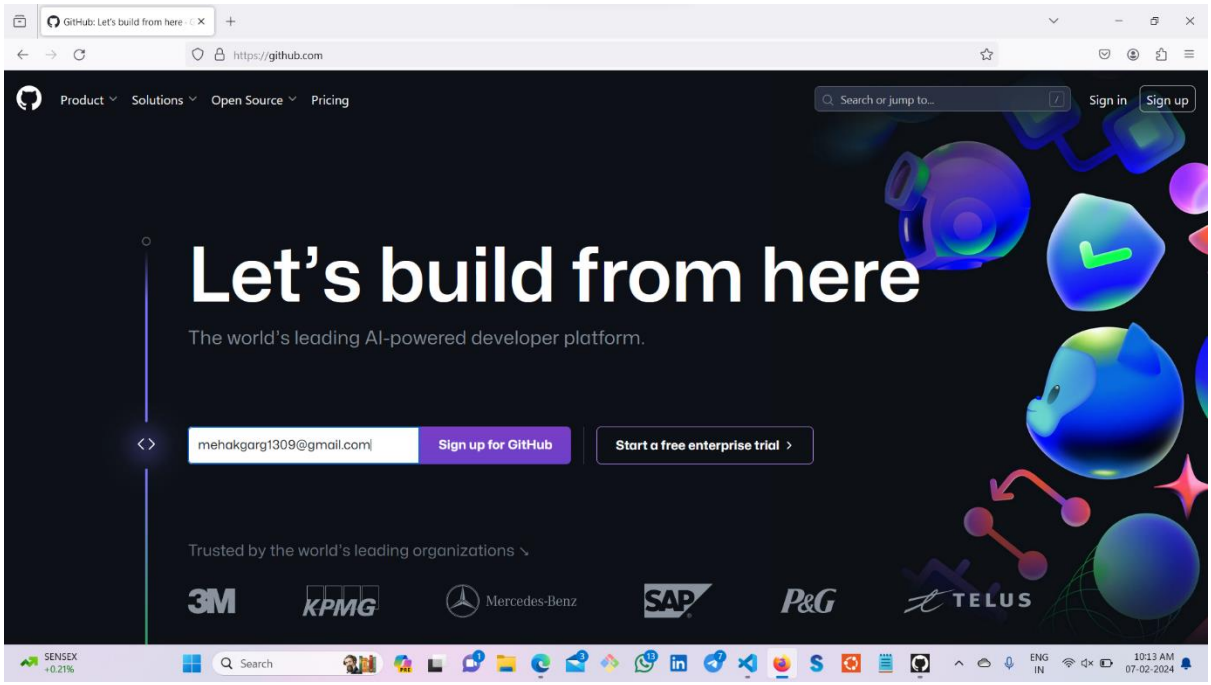
GitHub is a code hosting platform for version control and collaboration. GitHub is a development platform inspired by the way you work. From open source to business, we can host and review code, manage projects, and build software alongside 36 million developers.

Advantages:

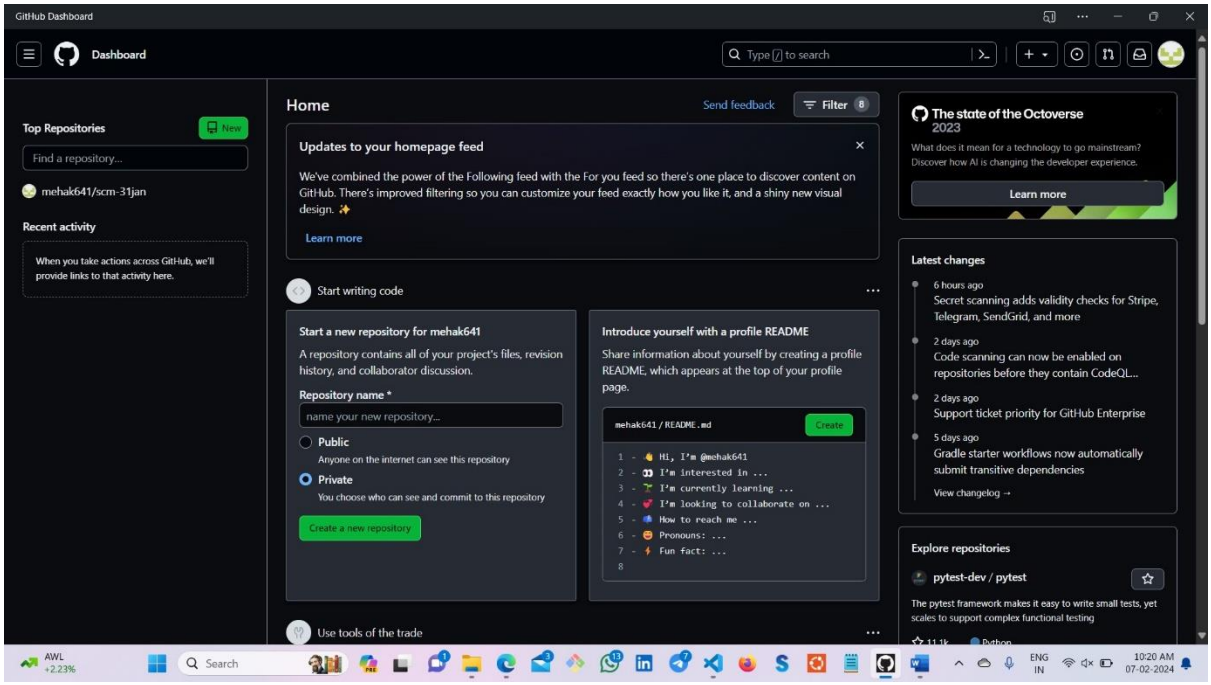
- Documentation.
- Showcase your work.
- Markdown.
- GitHub is a repository.
- Track changes in your code across versions.
- Integration options.

Procedure:

Search about GitHub: https://github.com/signup



By signing up for git you must remember your email and pass phases or password. For a new user, you must add your email and click on Sign up for GitHub. Otherwise click on Sign In at the top right corner.



For linking Git Hub with Git Bash:

Username:

git config --global user.name “username in github”

Email:

git config --global user.email “your email in github”

Check Username & Email:

git config user.name

git config user.email

```
Hp@HunarBhutani MINGW64 /e/scmFile
$ git config --global user.name "hunar"

Hp@HunarBhutani MINGW64 /e/scmFile
$ git config --global user.email "hunarbhutani1@gmail.com"

Hp@HunarBhutani MINGW64 /e/scmFile
$ git config user.name
hunar

Hp@HunarBhutani MINGW64 /e/scmFile
$ git config user.email
hunarbhutani1@gmail.com

Hp@HunarBhutani MINGW64 /e/scmFile
$
```

EXPERIMENT NO 3:-

Aim: Generate Logs on Git Hub

Theory:

Git Logs: The git log command shows a list of all the commits made to a repository. You can see the hash of each Git commit, the message associated with each commit, and more metadata. This command is basically used for displaying the history of a repository.

Why do we need logs?

Git log is a utility tool to review and read a history of everything that happens to a repository. Anything we change at what time, by which log, everything is getting recorded in git logs.

```
Hp@HunarBhutani MINGW64 /e/scmFile (mast
er)
$ cat file01.txt
hello guys
we are using git bash
Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ ls
contact.txt  file01.txt

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git add file01.txt

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git commit -m"commit of file01"
[master (root-commit) 93862f3] commit of file01
 2 files changed, 4 insertions(+)
 create mode 100644 contact.txt
 create mode 100644 file01.txt

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git status
On branch master
nothing to commit, working tree clean

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git log
commit 93862f3897593ebbd8ef45975b9db4d52905a5af (HEAD
r)
Author: hunar bhutani <hunarbhutani1@gmail.com>
Date:   Thu Feb 15 11:48:03 2024 +0530

    commit of file01

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$
```

You can use command **git log** to access logs(every change you make with time and date).

EXPERIMENT NO 4:-

Aim: Creating and Visualizing the Branches On Git Client

Theory:

How to create branches?

The main branch in which we are working is master branch. you can use the “git branch” command with the branch name and the commit SHA for the new branch.

1. For creating a new branch: git branch “name of the branch”.

```
    commit of file01

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git branch
* master

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$
```

2. To check how many branches we have:

```
Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git branch feature

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git branch
  feature
* master

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$
```

As you can see here one branch is showing that I create-feature.

3. To change the present working branch: git checkout “name of the branch” and command to go back to the master directory:

```
$ git branch feature

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git branch
  feature
* master

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git branch
  feature
* master

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git branch feature
fatal: a branch named 'feature' already exists

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git checkout feature
Switched to branch 'feature'

Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ git checkout master
Switched to branch 'master'

Hp@HunarBhutani MINGW64 /e/scmFile (master)
$
```

Here, you see by using checkout command we can switch branches and from branches to even master branch too.

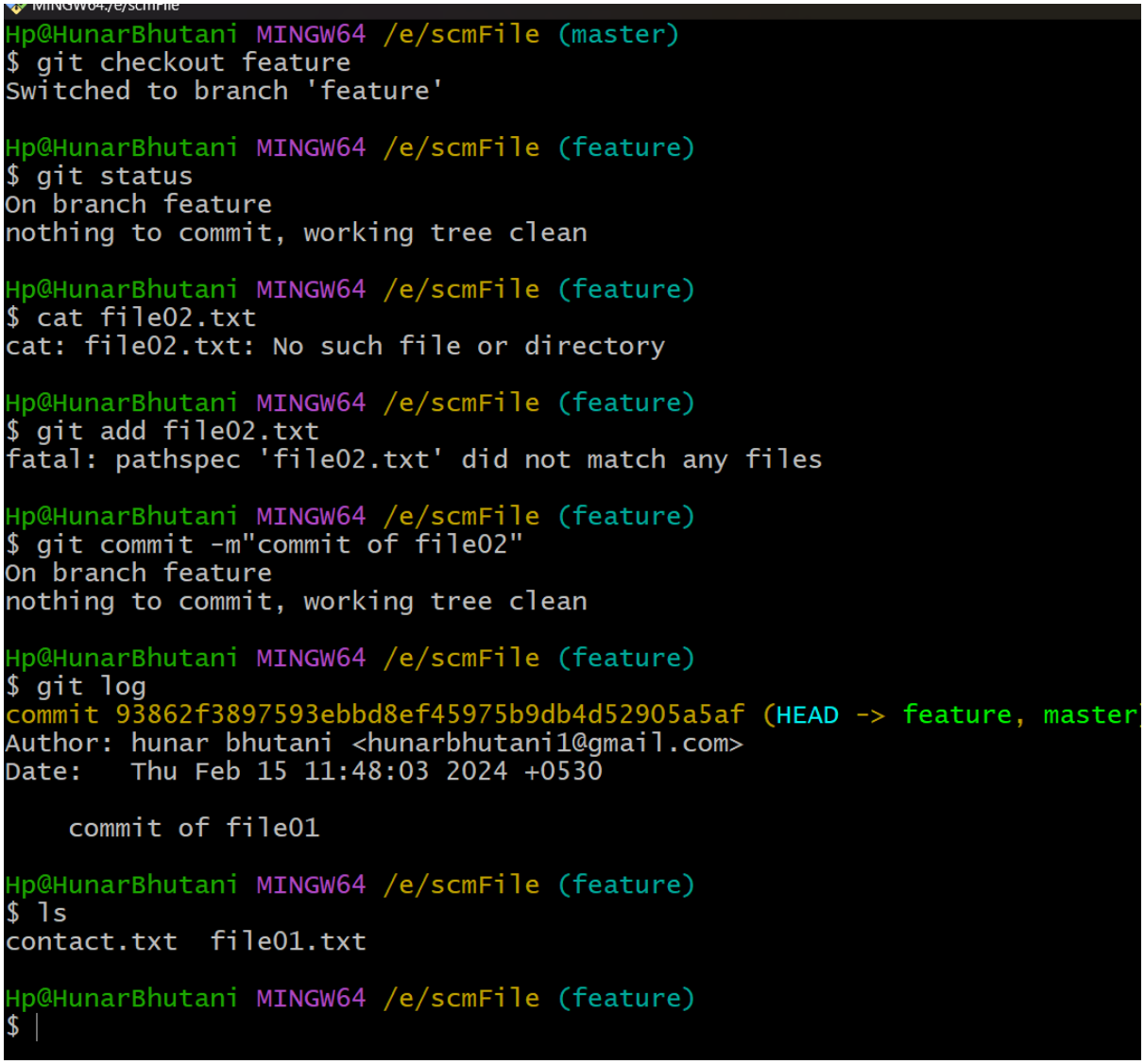
Visualizing branches:

for visualizing, we have to create a new file in the branch that we made “feature” instead of the master branch. After this we have to do three step architecture that is working directory, staging area and git repository.

Firstly I’ve changed the branch from master to feature that I previously made and after that I check git status. Now I add text in file02 file (file02.txt) and use git add “file_name”.

Then I use git commit -m “commit of file02” command for the changes I made and insertions I do.

At last I check my activities with log command.


Hp@HunarBhutani MINGW64 /e/scmFile (master)
$ git checkout feature
Switched to branch 'feature'
Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ git status
On branch feature
nothing to commit, working tree clean
Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ cat file02.txt
cat: file02.txt: No such file or directory
Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ git add file02.txt
fatal: pathspec 'file02.txt' did not match any files
Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ git commit -m"commit of file02"
On branch feature
nothing to commit, working tree clean
Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ git log
commit 93862f3897593ebbd8ef45975b9db4d52905a5af (HEAD -> feature, master
Author: hunar bhutani <hunarbhutani1@gmail.com>
Date: Thu Feb 15 11:48:03 2024 +0530
commit of file01
Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ ls
contact.txt file01.txt
Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$

EXPERIMENT NO 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:

When a project is residing in our local system we don't know whether the project is tracked by Git or not. In any of the case, this project directory is called our Working directory.

Staging Area:

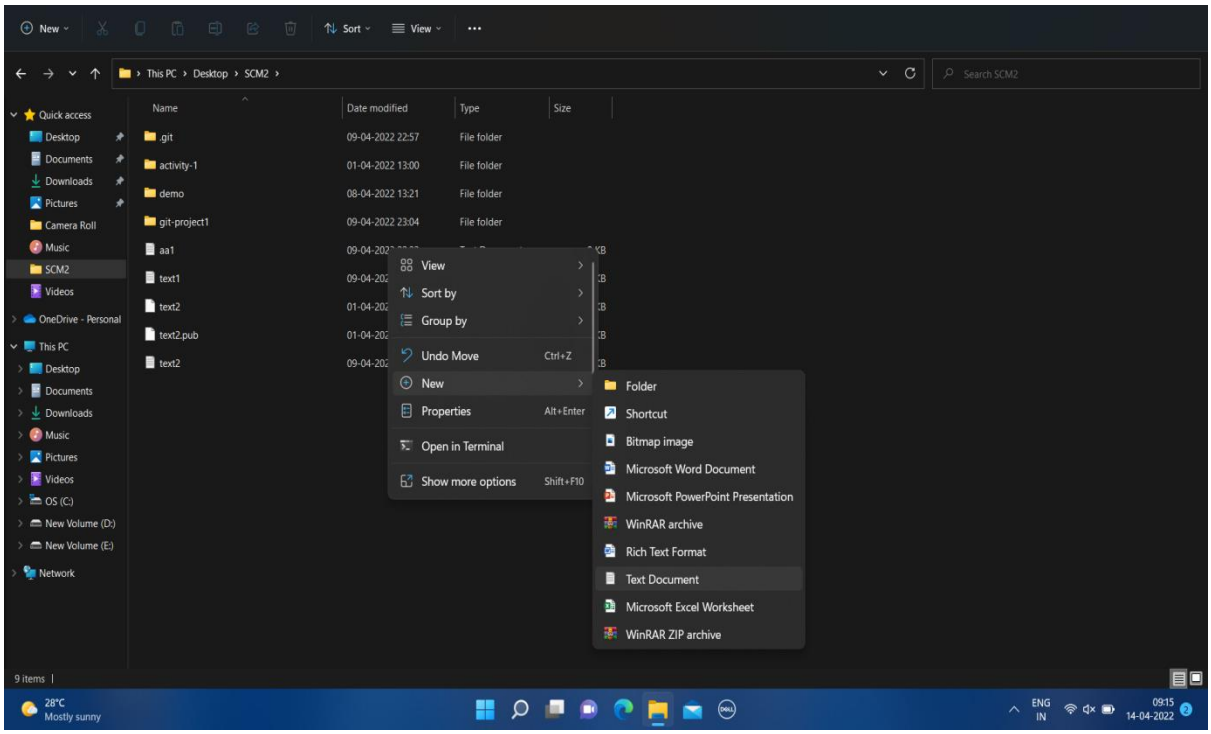
The staging area is like a rough draft space, it's where you can git add the version of a file or multiple files that you want to save in your next commit (in other words in the next version of your project)

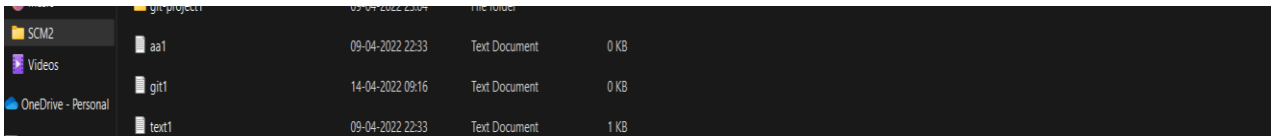
Git Directory:

The . git folder contains all information that is necessary for the project and all information relating commits, remote repository address, etc. It also contains a log that stores the commit history. This log can help you to roll back to the desired version of the code

Remote Repository: Remote repositories are hosted on a server that is accessible for all team members - most likely on the internet or on a local network. Assessable and reachable by all.

Screenshot:





```
Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ git branch beta

Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ pwd
/e/scmFile

Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ mkdir git-project

Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ git init
Reinitialized existing Git repository in E:/scmFile/.git/

Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ touch python.py

Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ vi python.py

Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ touch python

Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ vi python

Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ git add python.py
warning: in the working copy of 'python.py', LF will be replaced by CRL
e next time Git touches it

Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ git branch beta
fatal: a branch named 'beta' already exists

Hp@HunarBhutani MINGW64 /e/scmFile (feature)
$ git checkout beta
Switched to branch 'beta'
A       python.py

Hp@HunarBhutani MINGW64 /e/scmFile (beta)
$ git branch
* beta
  feature
  master
```