SOURCE CODE MANAGEMENT FILE

Submitted by:

Name: JANVI
Roll No.: 2310991854
Group 21-A

Submitted To:

Dr Renu Popli

Department of ComputerScience & EngineeringChitkara University Instituteof Engineering andTechnology

Rajpura,Punjab

CHITKARA UNIVERSITY

SourceCode ManagementFile

SubjectName: SourceCodeManagement(SCM)
SubjectCode:22CS003

Submitted by:

Name: JANVI
Roll No.: 2310991854
Group: 21-A

Task 1.1 Submission (Week 4)

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

Index

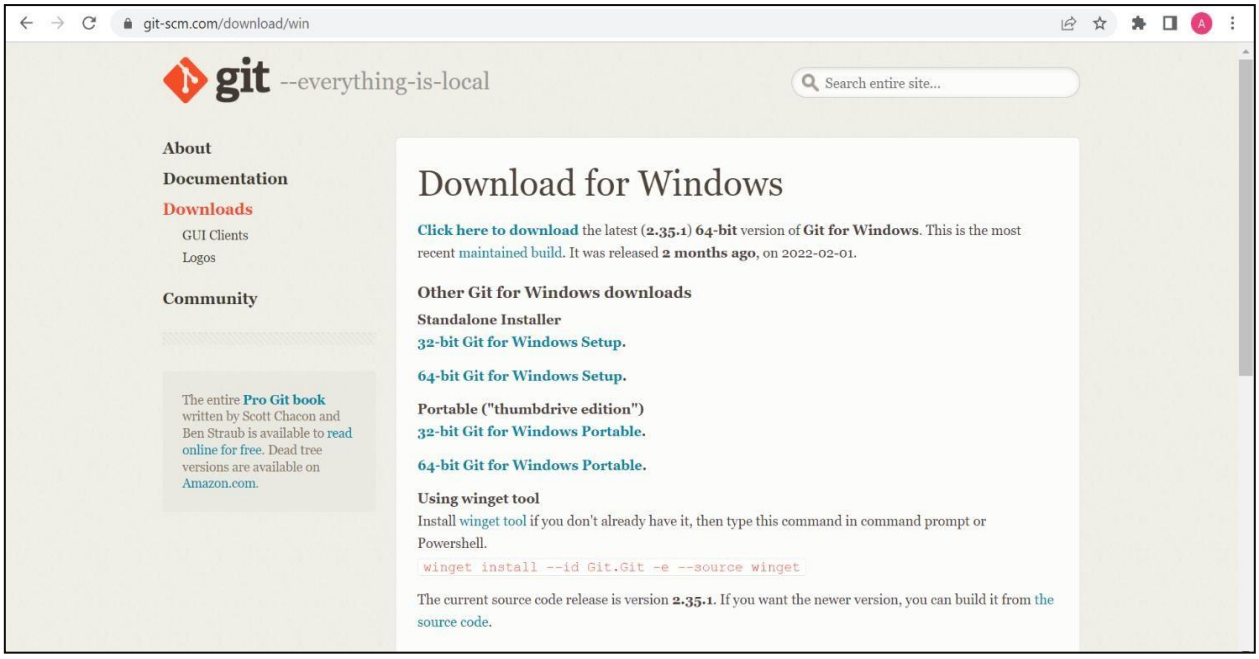
Experiment 1

Aim:
Setting up of Git Client.

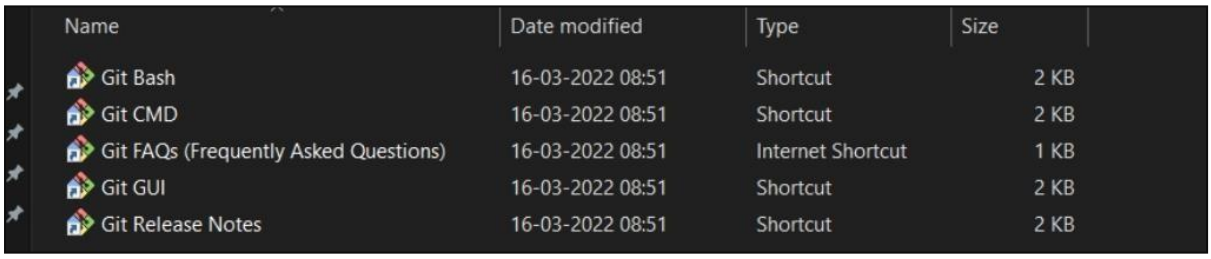
Theory:
Git is a distributed version control system that revolutionized collaborative software development. Developed by Linus Torvalds in 2005, Git allows multiple developers to work on a project simultaneously without conflicts, enabling efficient and seamless coordination. It tracks changes in source code, managing versions and facilitating collaboration among team members. Git's decentralized nature means that each user has a complete repository on their local machine, enhancing flexibility and enabling offline work. Its versatility extends beyond code, making it an essential tool for tracking changes in any type of text-based document. As an open-source tool, Git has become a cornerstone in modern software development, contributing to streamlined workflows and enhanced code quality.

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 the platform and bit-version or IOS it will start downloading automatically.

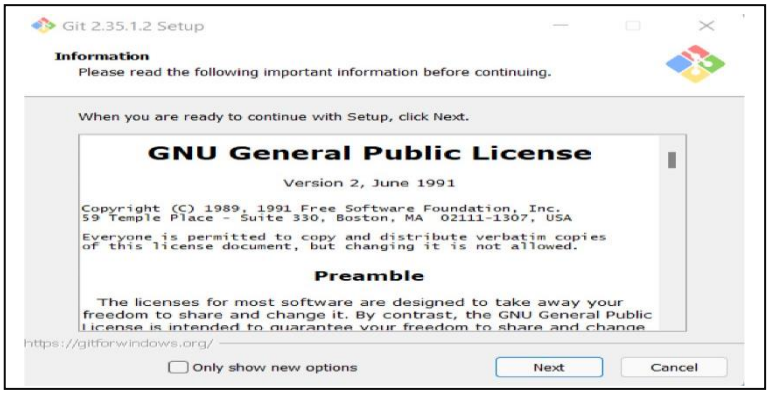
Snapshots of download:



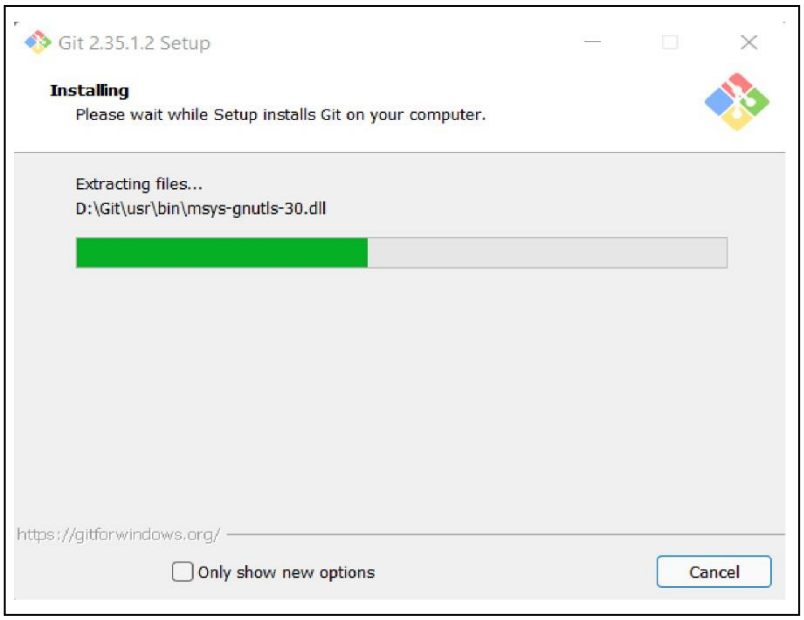
Opted for 64-bit Git for Windows Setup



Git and its files in downloads



Git Setup



Git Installation

Git-Bash launched

Experiment – 2

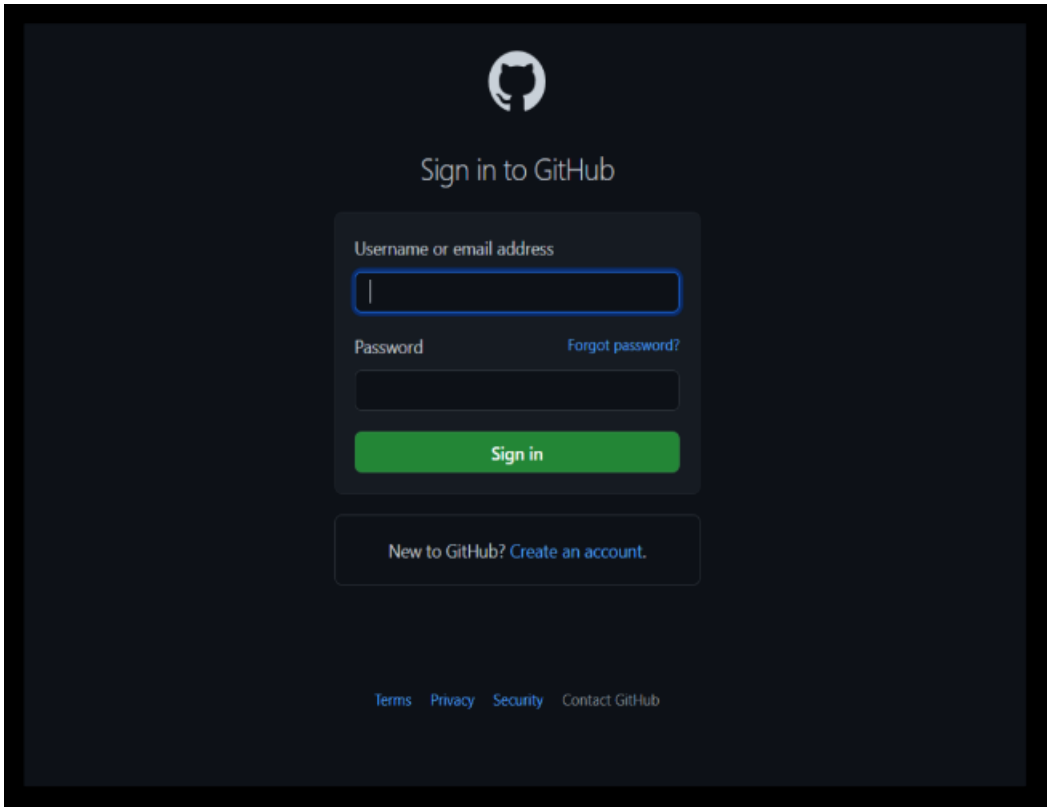
Aim:
Setting up of GitHub account

Theory:
GitHub is a web-based platform that serves as a central hub for software development collaboration using Git, a distributed version control system. It enables developers to host their code repositories, manage project workflows, and facilitate collaboration among team members or the wider community. GitHub provides features such as issue tracking, pull requests, code review, and wikis, which streamline the development process and foster open communication. With its user-friendly interface and extensive integration capabilities, GitHub has become the go-to platform for individual developers, open-source projects, and enterprises alike to manage their codebases efficiently and transparently. Additionally, GitHub fosters a vibrant ecosystem of developers sharing their work, contributing to open-source projects, and collaborating on innovative solutions across a myriad of domains, making it a cornerstone of modern software development practices.

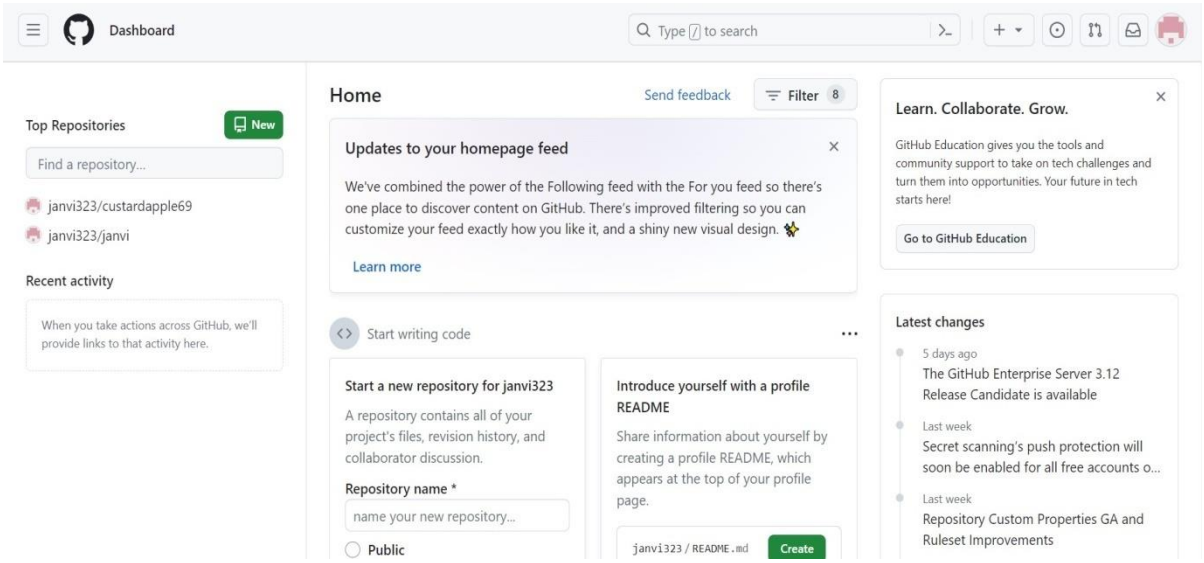
Procedure:
To make an account on GitHub, we search for GitHub on our browser or visithttps://github.com/signup. Then, we will enter our mail ID and create ausername and password for a GitHub account.

Snapshots:

After visiting the link this type of interface will appear, we can sign in or sign up accordingly.



If we have an account already, we can click on sign in and enter our credentials.



GitHub interface of our account.

Experiment – 3

Aim:
Generating logs.

Theory:
Generating logs in Git is about generating a history of whatever we did in our git repository, using the 'git log' command in the git bash terminal. Logs can be used to cross check the changes we did and what we want to do further based on the previous changes. It also contains the number of insertions and deletions and the time of happening also. The git log command has further variations in it like 'git log –p –n' to check a specific number of previous commits/changes where n is a positive integer.

Procedure:
To use git bash and its commands, first we have to initialize git in our work folder. This is done by first opening git bash in the folder by right clicking and clicking on 'open git bash here' from the dropdown menu. A git bash terminal is opened. Now using the 'git init' command, a '.git' folder is created in the work folder.

When we are using git for the first time, we have to configure git using certain commands. We have to give a username and email which will be visible to the people when we upload our projects or make changes in the existing ones.

For this purpose we use:

"git config –global user.name yourname "
"git config –global user.emailyouremail "

To verify our credentials, we can use:

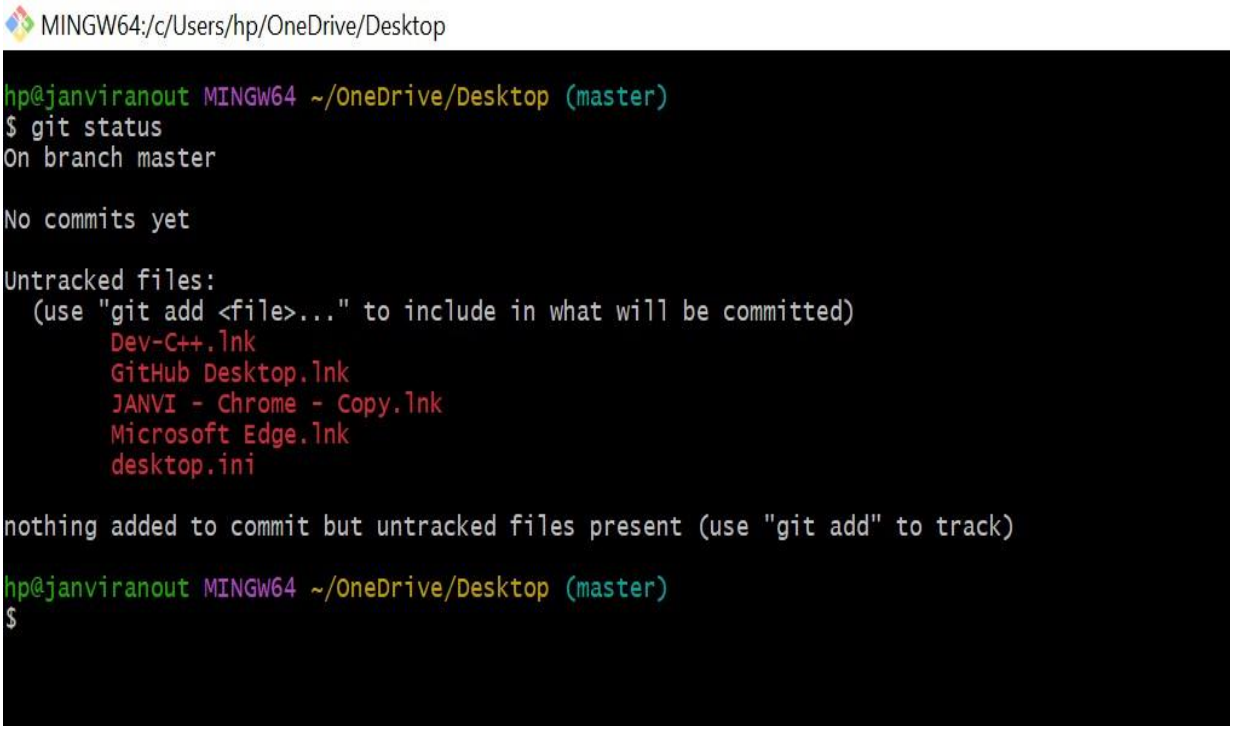
"git config –global user.name"

"git config –global user.email"

Some Important Commands

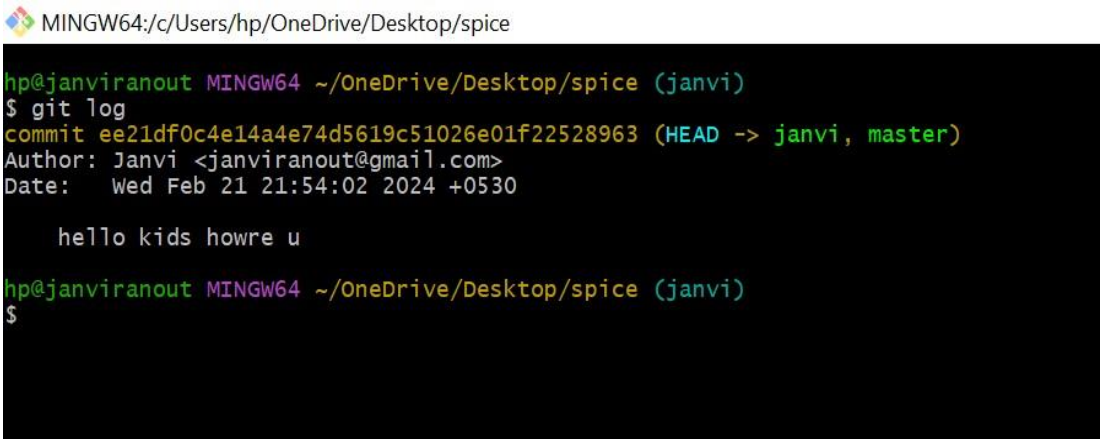
- ls - gives the list of items in the work folder.
- ls –lart – gives the hidden files also.
- git status – displays the state of the working directory and the staged snapshot.
- pwd – tells about the present working directory.
- cd – change directory.
- mkdir – make directory.
- touch – to create a file.
- clear – to clear the terminal.
- rm –rf .git – it removes the repository.
- git log – displays the history of commits in a repository.
- git diff – compares the working tree with staging area.
- git add – add a file to the staging area.
- git commit – to commit the changes to the remote server.

git status:



Currently there were no changes made in the directory, therefore it is showing working tree clean otherwise it shows the status of the untracked files, as well as the files not committed.

git log:



The ‘git log’ command displays a record of the commits in our Git repository. We can write ‘git log –p –n’ to check a specific number of previous commits, where n is a number.

Experiment – 4

Aim:
Create and visualize branches.

Theory:
In Git, branches are essentially separate lines of development that allow you to work on different features, bug fixes, or experiments without affecting the main codebase. Here's an overview of how branches work in Git:

1. **Master/Branch:** The default branch in a Git repository is often called master. This is where the main codebase resides and is typically considered the stable version of the project.

2. **Creating Branches**: You can create a new branch at any point in your Git repository's history. This new branch starts as an exact copy of the branch you were on when you created it. You can create a branch using the git branch <branch_name> command.

3. **Switching Branches**: You can switch between branches using the git checkout <branch_name> command. This allows you to start working on a different line of development.

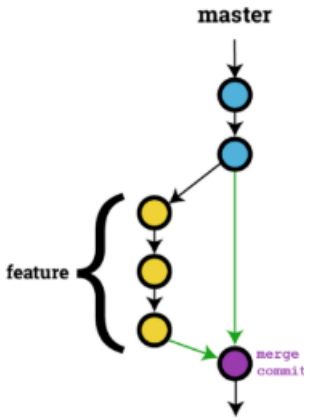
4. **Committing Changes:** Once you've made changes on a branch, you commit those changes using the git commit command. This records the changes you've made in the repository's history.

5. **Merging Branches:** After you've completed work on a branch and are satisfied with the changes, you can merge it back into another branch (often master). This combines the changes from the source branch into the target branch. You can do this with the git merge <branch_name> command.

6. **Branch Management**: Git provides commands to manage branches such as git branch -d <branch_name> to delete a branch once it's no longer needed, and git branch -m <new_branch_name> to rename a branch.

7. **Feature Branch Workflow:** One common branching strategy is the feature branch workflow, where each new feature or task is developed in its own branch, and then merged back into the main branch (master or main) when complete. This keeps the main branch stable while allowing for concurrent development of multiple features.

Understanding how branches work in Git is crucial for effective collaboration and version control in software development projects.

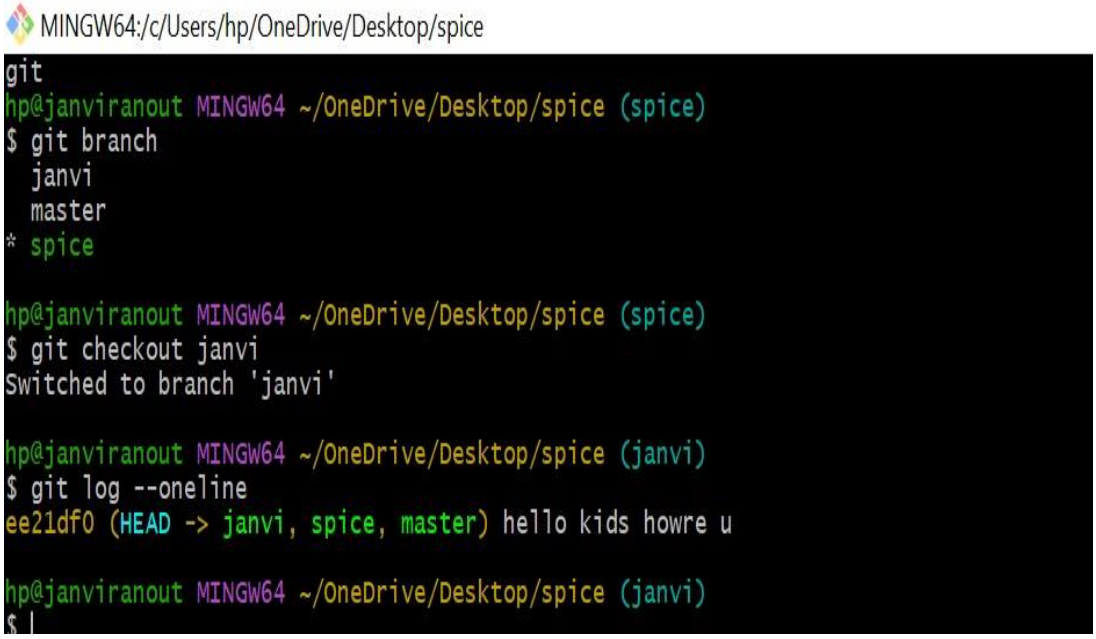


Branches in Git

Snapshots:

'git branch' command – To display the total branches available. Default branch is the master branch.

Adding and switching to a branch – branch1



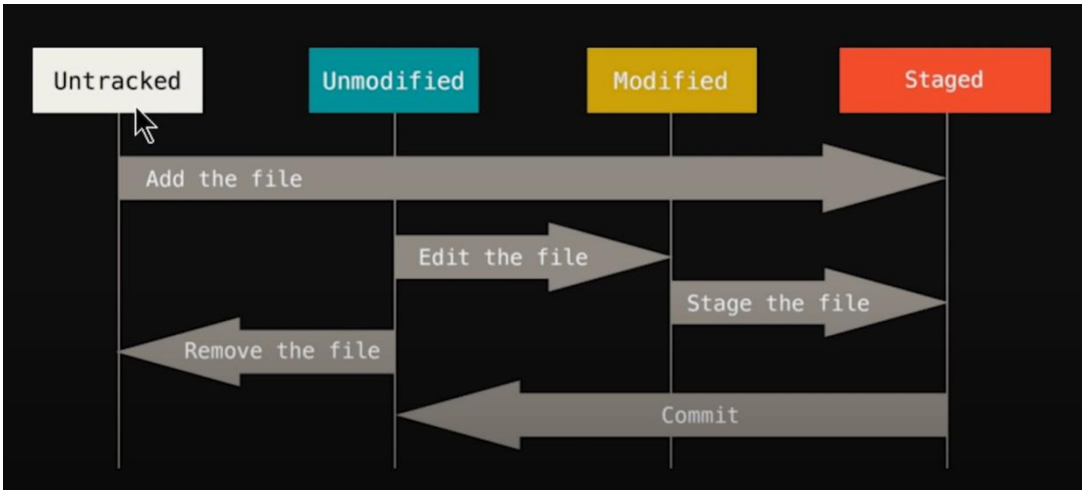
Checking commits in the branch made

Experiment – 5

Aim:
Git lifecycle description

Theory:
The lifecycle of a Git repository typically involves several stages as files are created, modified, and managed within the repository. Here's a general description of the Git lifecycle:

1. Working directory
2. Staging area
3. Git directory

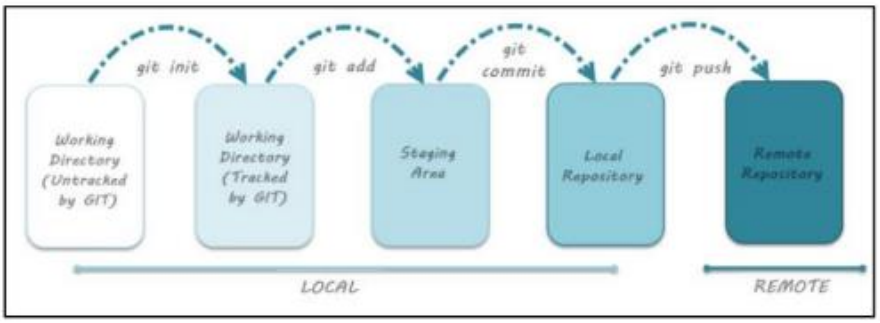


Working directory:
Any file residing in our local system which may or may not be tracked by git is said to be in the working directory.

Staging area:
Staging area is the virtual space where we bring our files before committing.

Git directory:
Now that the files to be committed are grouped and ready in the staging area, wecan commit these files. So, we commit this group of files along with a commitmessage explaining what is the commit about. Apart from commit message, thisstep also records the author and time of the commit. Now, a snapshot of the filesin the commit is recorded by Git. The information related to this commit is storedin the Git directory.

Remote Repository: It means mirror or clone of the local Git repository inGitHub. And pushing means uploading the commits from local Git repository toremote repository hosted in GitHub.



MINGW64:/c/Users/hp/OneDrive/Desktop/spice

```
g
hp@janviranout MINGW64 ~/OneDrive/Desktop/spice (janvi)
$ git init
Initialized empty Git repository in C:/Users/hp/OneDrive/Desktop/spice/.git/

hp@janviranout MINGW64 ~/OneDrive/Desktop/spice (master)
$ git add .

hp@janviranout MINGW64 ~/OneDrive/Desktop/spice (master)
$ git status
On branch master

No commits yet

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


hp@janviranout MINGW64 ~/OneDrive/Desktop/spice (master)
$ git commit -a -m " added a file named janvi.txt"
[master (root-commit) 963038f]  added a file named janvi.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 spice

hp@janviranout MINGW64 ~/OneDrive/Desktop/spice (master)
$ git status
On branch master
nothing to commit, working tree clean
```