CHITKARA UNIVERSITY

SOURCE CODE MANAGEMENT FILE

Submitted By:

Name: Jay Soni

Roll no.: 2310991943

Group: G-22(A)

Submission of:

Task 1.1

Submitted To:

Dr. Sharad

Professor

Chitkara University,

Rajpura

Dell

Source Code Management File

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

Subject Code: **CS181**

Cluster: **Beta**

Submitted By:

Name: **Jay Soni**

Roll No. **2310991943**

Group: **G-22(A)**

Task 1.1 Submission

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

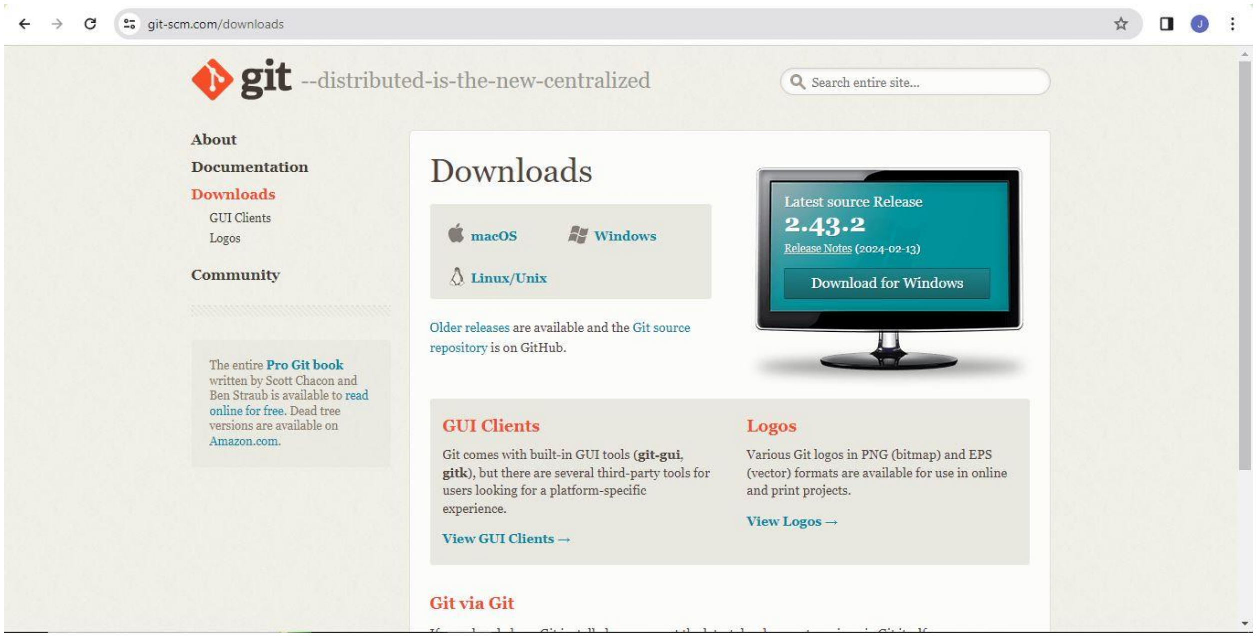
Experiment 1

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:



Downloads

Latest source Release
2.43.2
Release Notes (2024-02-13)

Older releases are available and the Git source repository is on GitHub.

The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

GUI Clients

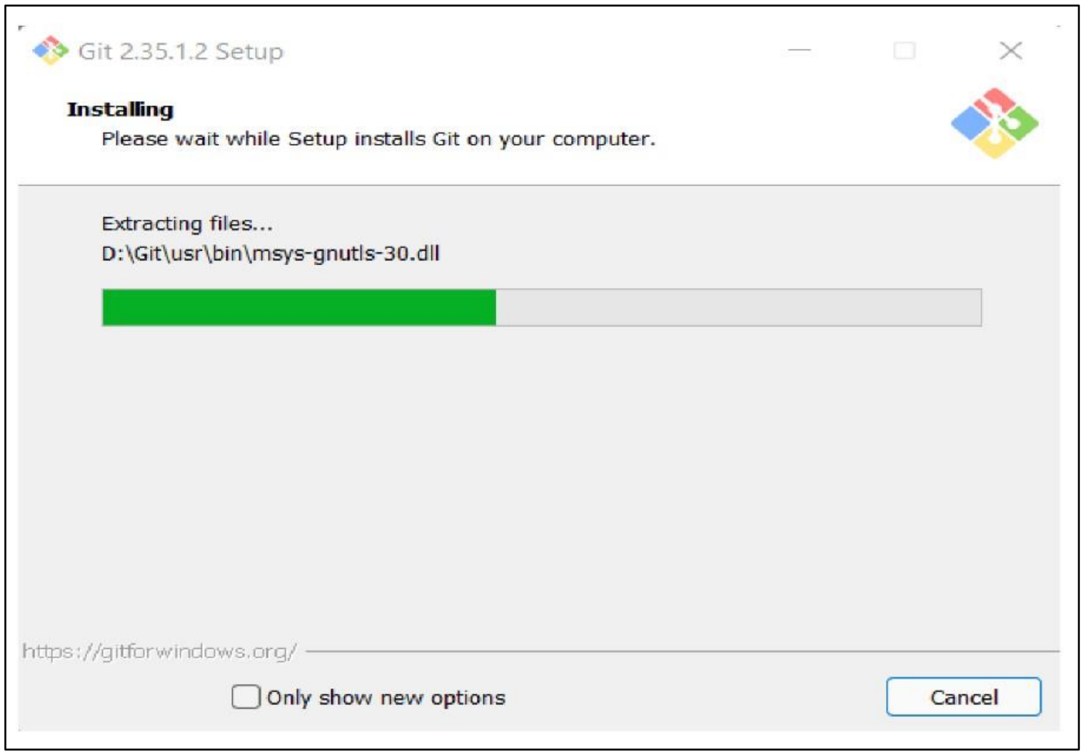
Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

Logos

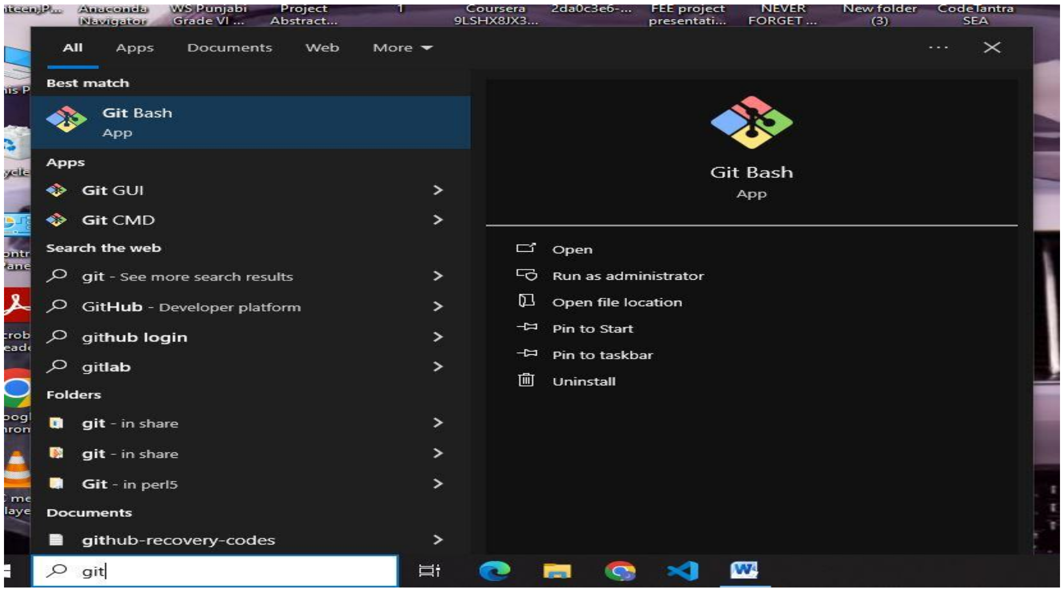
Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

Git via Git

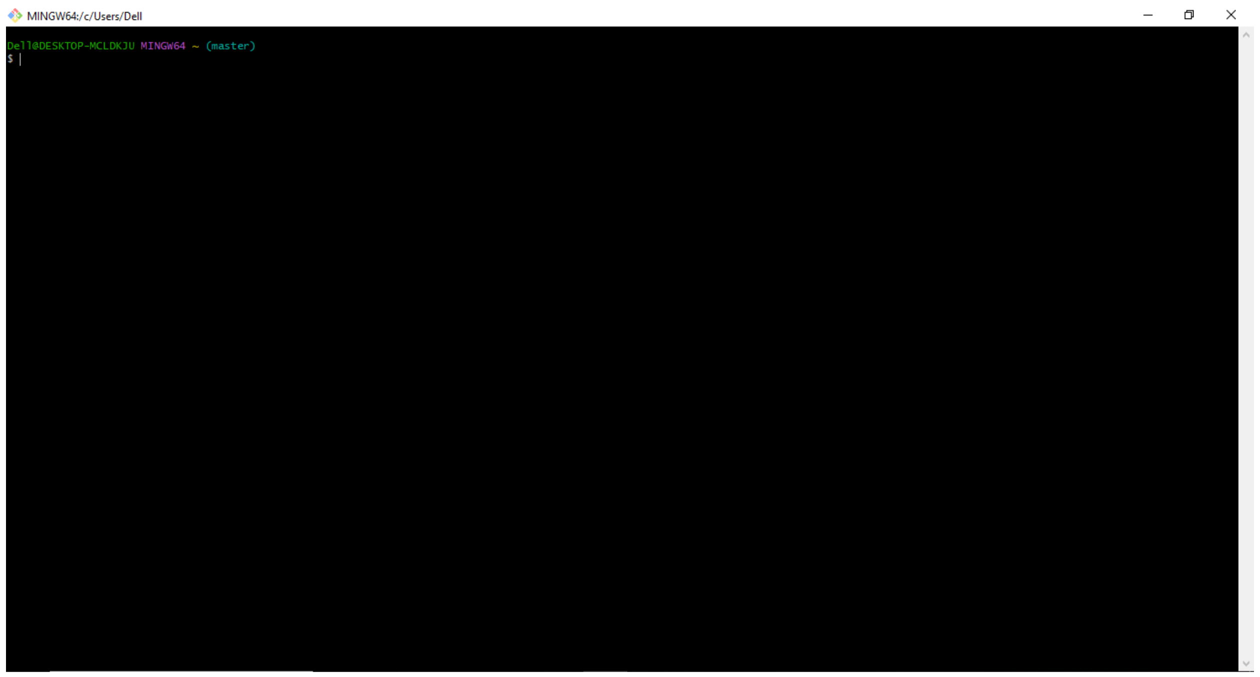
Opted for "64 bit git for windows setup"


Git 2.35.1.2 Setup
Installing
Please wait while Setup installs Git on your computer.
Extracting files...
D:\Git\usr\bin\msys-gnutls-30.dll
https://gitforwindows.org/
Only show new options
Cancel

Git Installation



Git Bash App in system



```
Dell@DESKTOP-MCLDKJU MINGW64 ~ (master)
$ 
```

Git Bash Launched

Experiment 2

Aim: Setting up GitHub Account

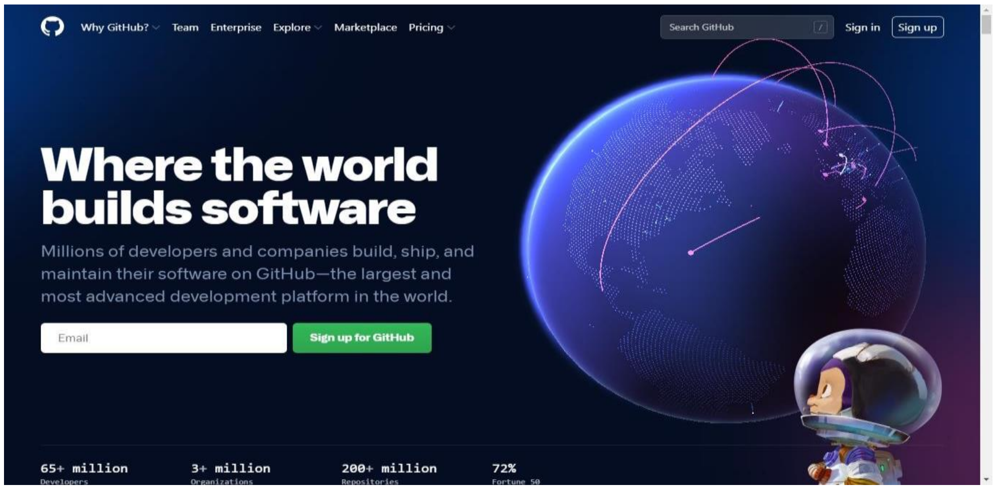
Theory:

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 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:



Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

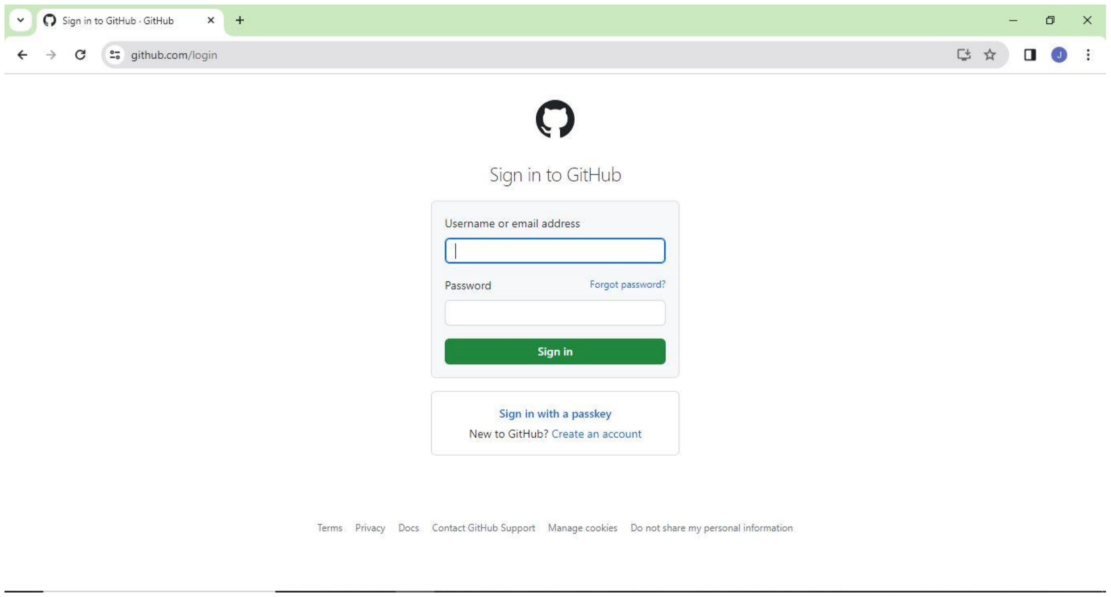
65+ million Developers

3+ million Organizations

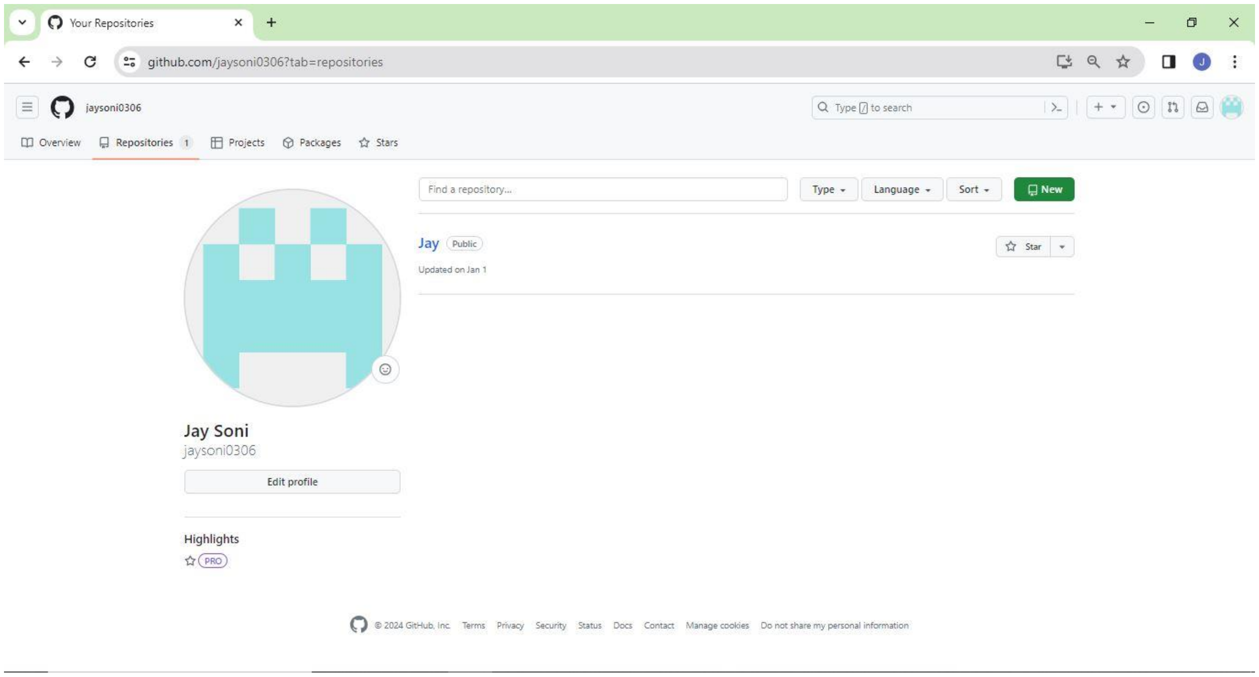
200+ million Repositories

72% Fortune 50

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.


Sign in to GitHub · GitHub
github.com/login
Sign in to GitHub
Username or email address
Password
Forgot password?
Sign in
Sign in with a passkey
New to GitHub? Create an account
Terms Privacy Docs Contact GitHub Support Manage cookies Do not share my personal information

Github Login



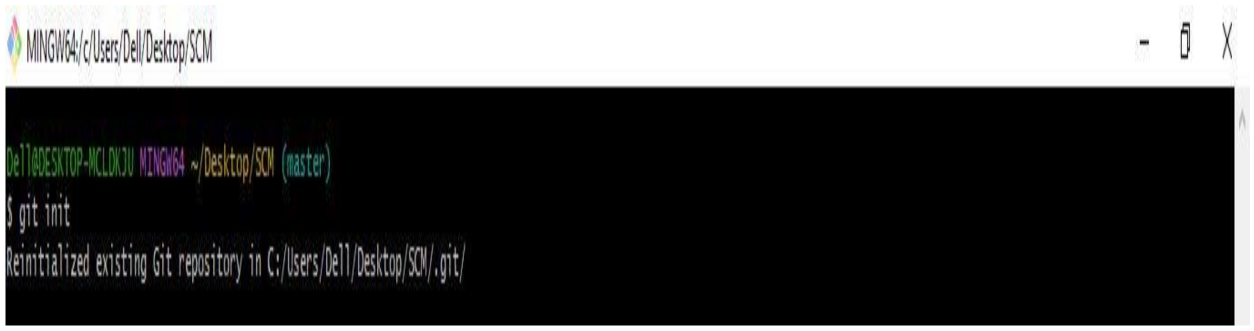
Github interface

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 folder ".git".



```
Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git init
Reinitialized existing Git repository in C:/Users/Dell/Desktop/SCM/.git/
```

git init

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:

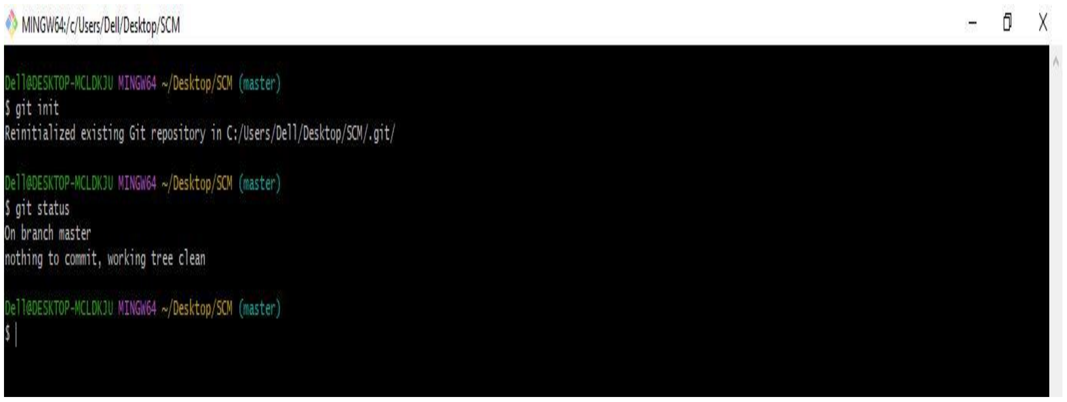
"git config --global user.name Name"
"git config --global user.email email"

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

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

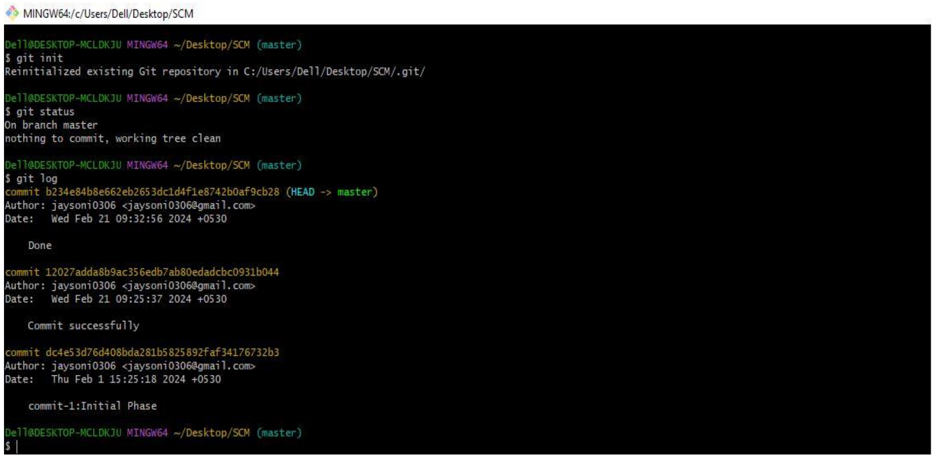
Some Important Commands

- 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.



```
MINGW64:/c/Users/Dell/Desktop/SCM

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git init
Reinitialized existing Git repository in C:/Users/Dell/Desktop/SCM/.git/

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git status
On branch master
nothing to commit, working tree clean

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ |
```

git status



```
MINGW64:/c/Users/Dell/Desktop/SCM

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git init
Reinitialized existing Git repository in C:/Users/Dell/Desktop/SCM/.git/

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git status
On branch master
nothing to commit, working tree clean

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git log
commit b234e84b8e662eb2653dc1d4f1e8742b0af9cb28 (HEAD -> master)
Author: jaysoni0306 <jaysoni0306@gmail.com>
Date:   Wed Feb 21 09:32:56 2024 +0530

    Done

commit 12027adda8b9ac356edb7ab80edadcbc0931b044
Author: jaysoni0306 <jaysoni0306@gmail.com>
Date:   Wed Feb 21 09:25:37 2024 +0530

    Commit successfully

commit dc4e53d76d408bda281b5825892faf34176732b3
Author: jaysoni0306 <jaysoni0306@gmail.com>
Date:   Thu Feb 1 15:25:18 2024 +0530

    commit-1:Initial Phase

Dell@DESKTOP-MCLDKJU MINGW64 ~/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 metadata.

Experiment 4

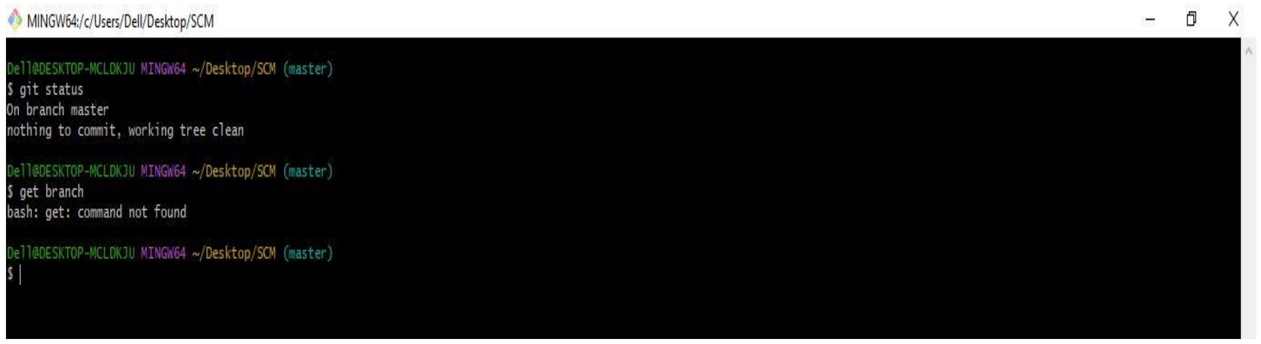
Aim: Create and visualize branches

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 master branch.

Create branches: The master branch in git is called as master branch. But we can make branches out of this master 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:

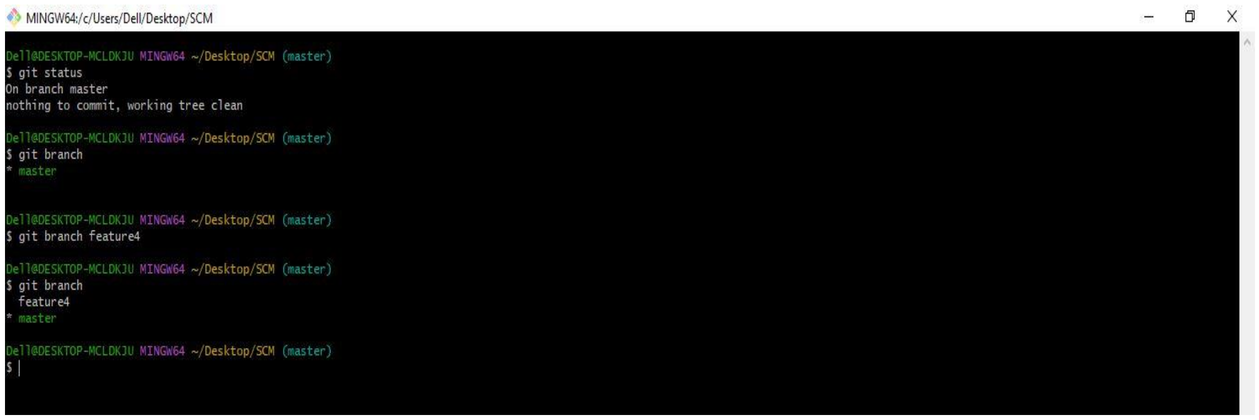
For creating a new branch, git branch name by default is master branch.

Snapshots–


MINGW64:/c/Users/Dell/Desktop/SCM

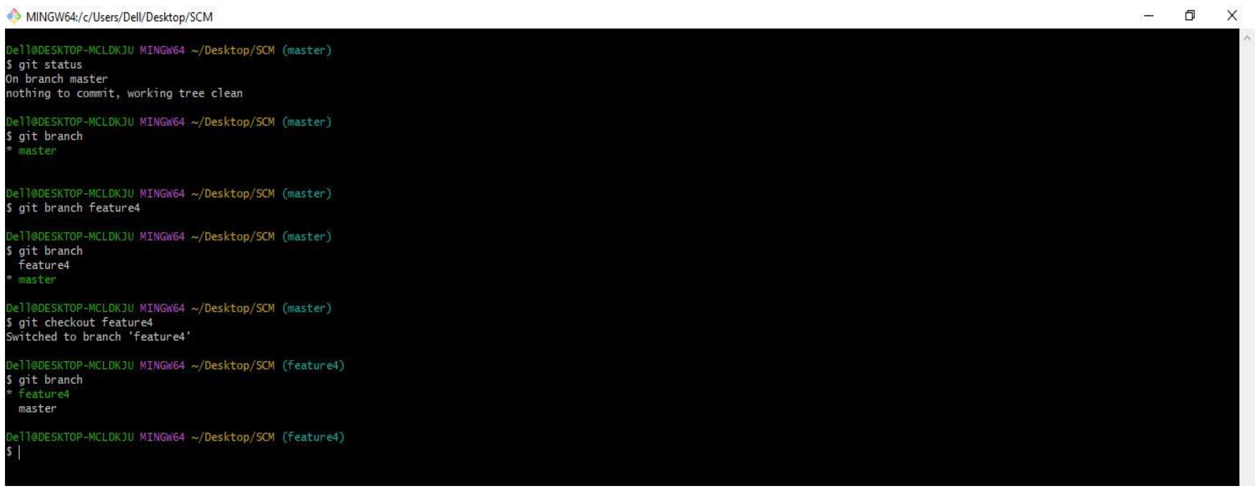
```
Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git status
On branch master
nothing to commit, working tree clean

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ get branch
bash: get: command not found

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ |
```

Default branch is master branch.



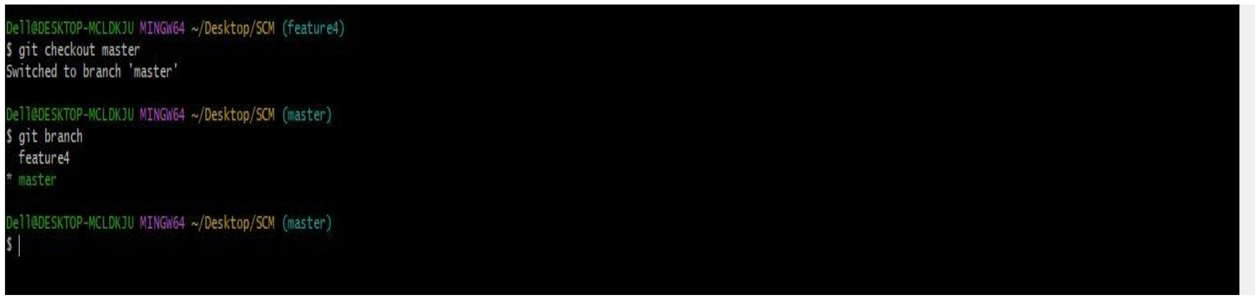
```
MINGW64:/c/Users/Dell/Desktop/SCM

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git status
On branch master
nothing to commit, working tree clean

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git branch
* master


Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git branch feature4

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git branch
  feature4
* master

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ |
```

Adding a feature branch



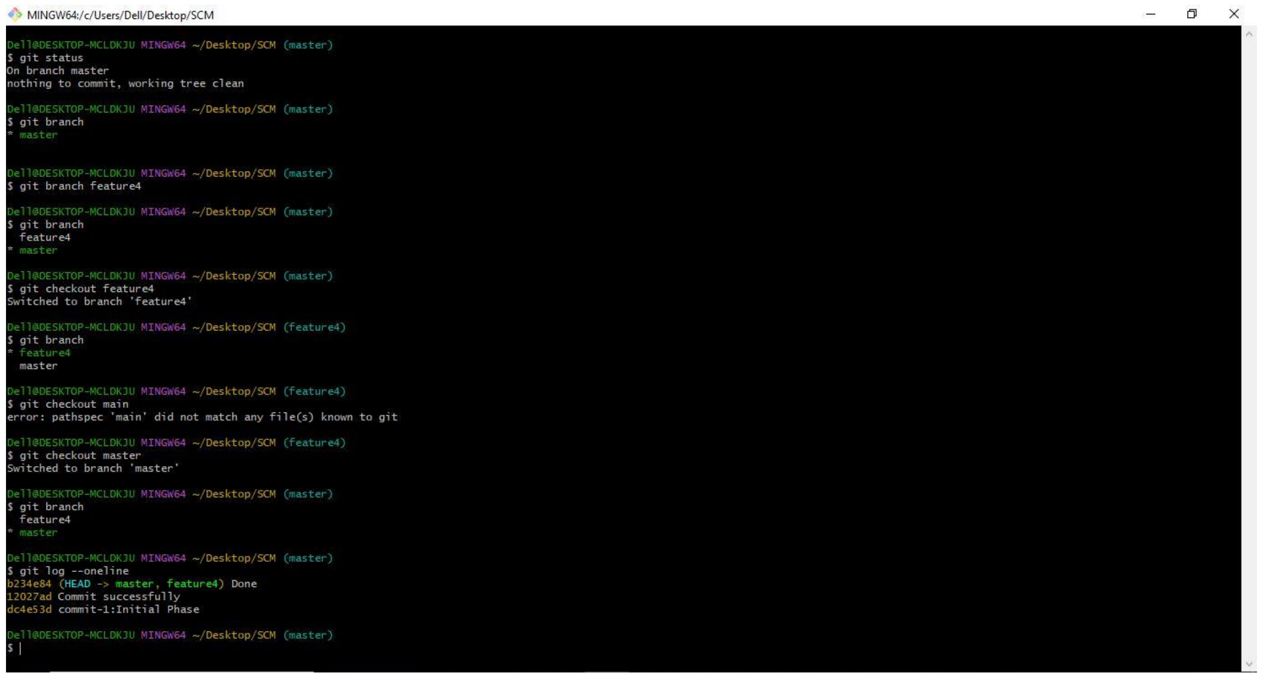
```
MINGW64:/c/Users/Dell/Desktop/SCM

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git status
On branch master
nothing to commit, working tree clean

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git branch
* master


Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git branch feature4

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git branch
  feature4
* master

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git checkout feature4
Switched to branch 'feature4'

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (feature4)
$ git branch
* feature4
  master

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (feature4)
$ |
```

Switching to feature branch



```
Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (feature4)
$ git checkout master
Switched to branch 'master'

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git branch
  feature4
* master

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ |
```

Switching to master branch



```
Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git status
On branch master
nothing to commit, working tree clean

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git branch
* master


Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git branch feature4

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git branch
  feature4
* master

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git checkout feature4
Switched to branch 'feature4'

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (feature4)
$ git branch
* feature4
  master

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (feature4)
$ git checkout main
error: pathspec 'main' did not match any file(s) known to git

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (feature4)
$ git checkout master
Switched to branch 'master'

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git branch
  feature4
* master

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ git log --oneline
b234e84 (HEAD -> master, feature4) Done
12027ad Commit successfully
dc4e53d commit-1:Initial Phase

Dell@DESKTOP-MCLDKJU MINGW64 ~/Desktop/SCM (master)
$ |
```

Checking commits

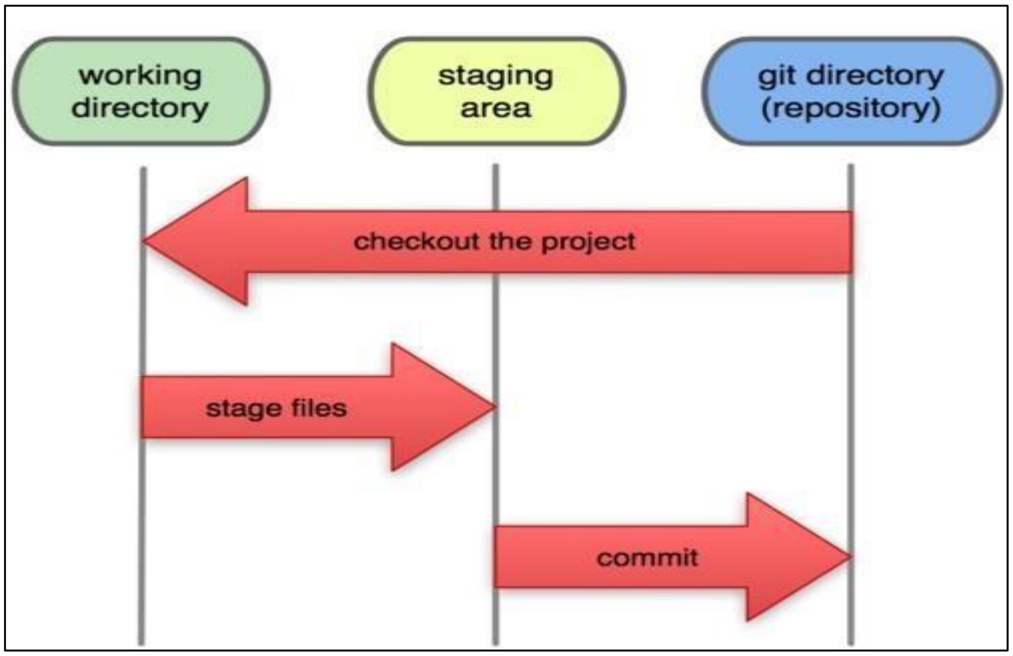
Experiment 5

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. In either case, this project directory is called your Working directory.

Staging Area:

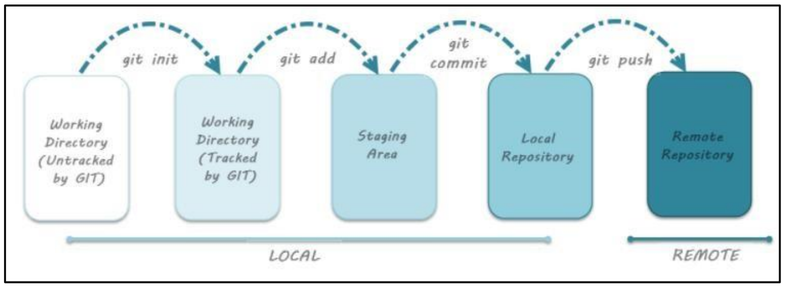
Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.

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.

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.


git init
git add
git commit
git push
Working Directory (Untracked by GIT)
Working Directory (Tracked by GIT)
Staging Area
Local Repository
Remote Repository
LOCAL
REMOTE