Source Code Management

Submitted by:
Amarbir Singh
2110990159
G8

Submitted to:
Dr . Monit Kapoor

1	Version control with Git	1-13
2	Project	13-17
3	Task 1.2	18-24

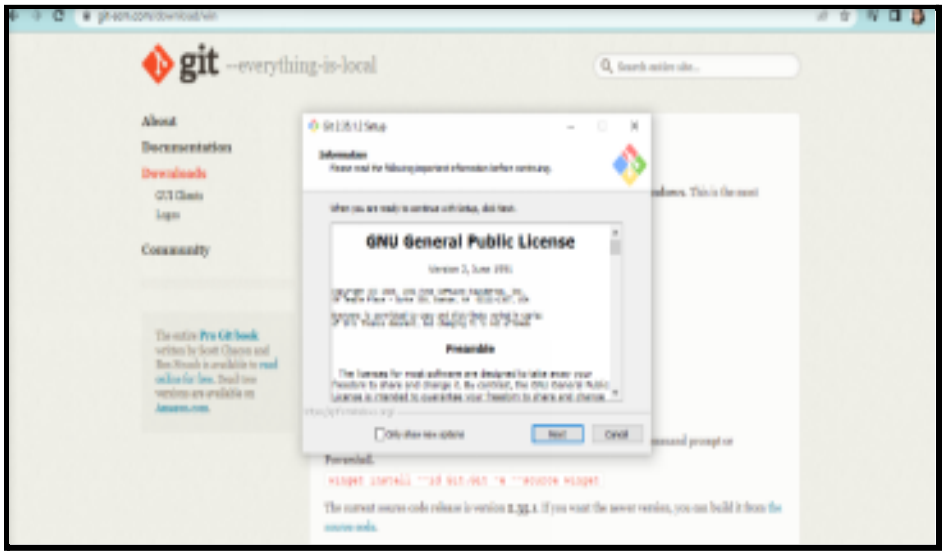
Step 1 - Installing Git:
Before we start using Git, we have to make it available on your computer. Even if it's already
Installed , it's probably a good idea to update to the latest version. We can either install it as a package or via another installer, or download the source code and compile it yourself.

Step 2 - Installing on windows:
There are also a few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to https://git-scm.com/download/win and the download will start automatically.

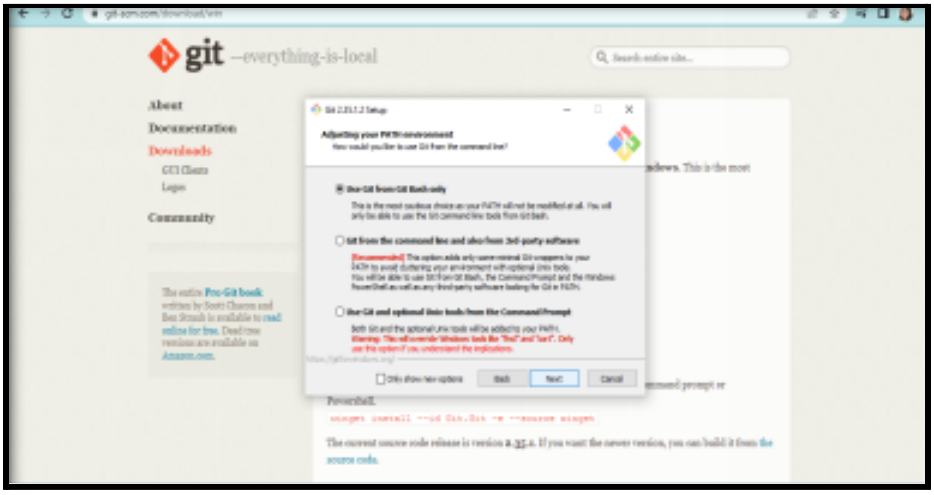
Note that this is a project called Git for Windows, which is separate from Git itself; for more information on it, go to https://gitforwindows.org.

1.

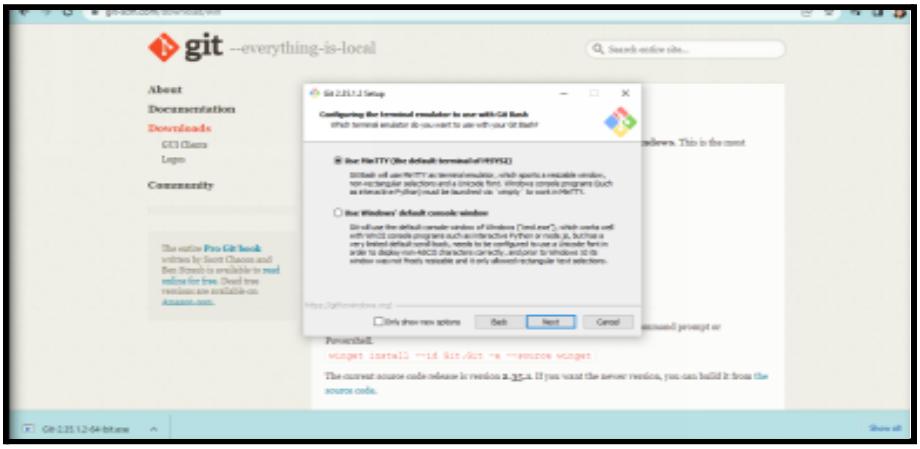
2.



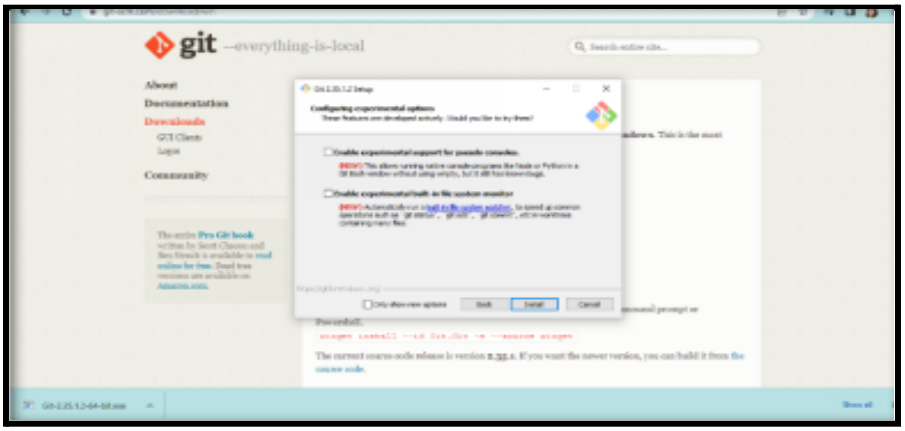
3.


git --everything-is-local
About
Documentation
Downloads
GUI Clients
Logos
Community

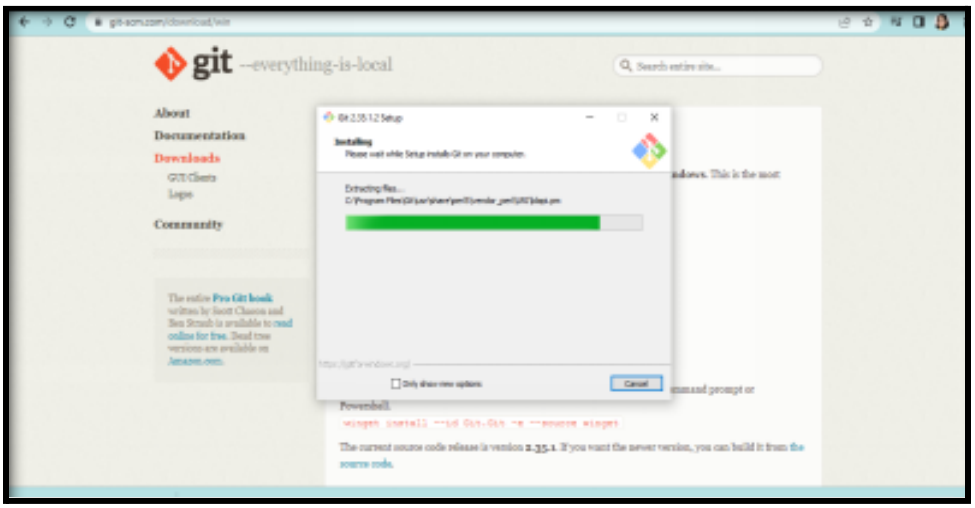
4.


git --everything-is-local
About
Documentation
Downloads
GUI Clients
Logos
Community

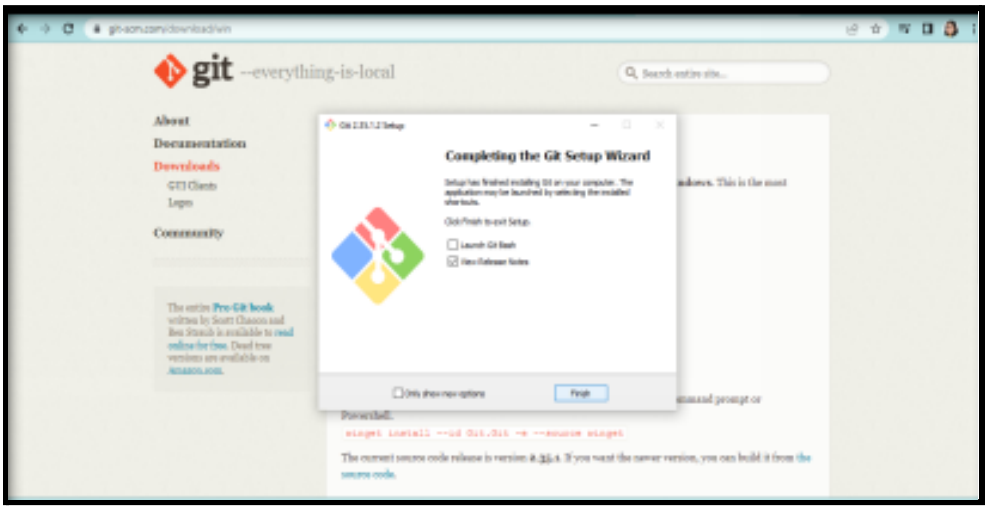
5.


git --everything-is-local
About
Documentation
Downloads
GUI Clients
Logos
Community

6.

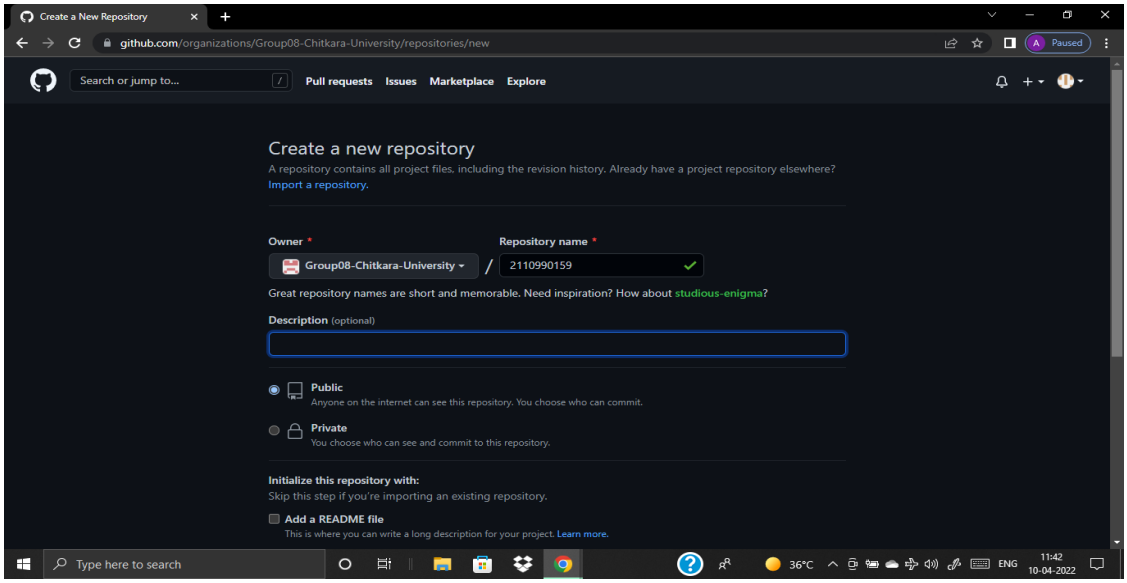

git --everything-is-local
About
Documentation
Downloads
GUI Clients
Logos
Community
Cancel

7.

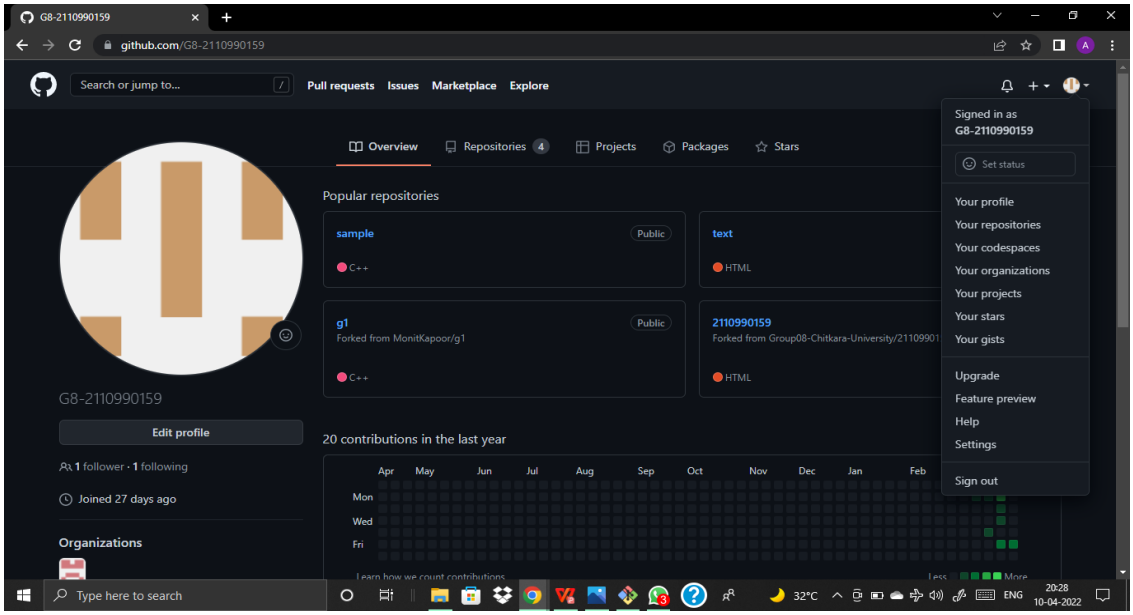

git --everything-is-local
About
Documentation
Downloads
GUI Clients
Logos
Community
Completing the Git Setup Wizard
Finish

Step 3 – Setting up github account:

1.



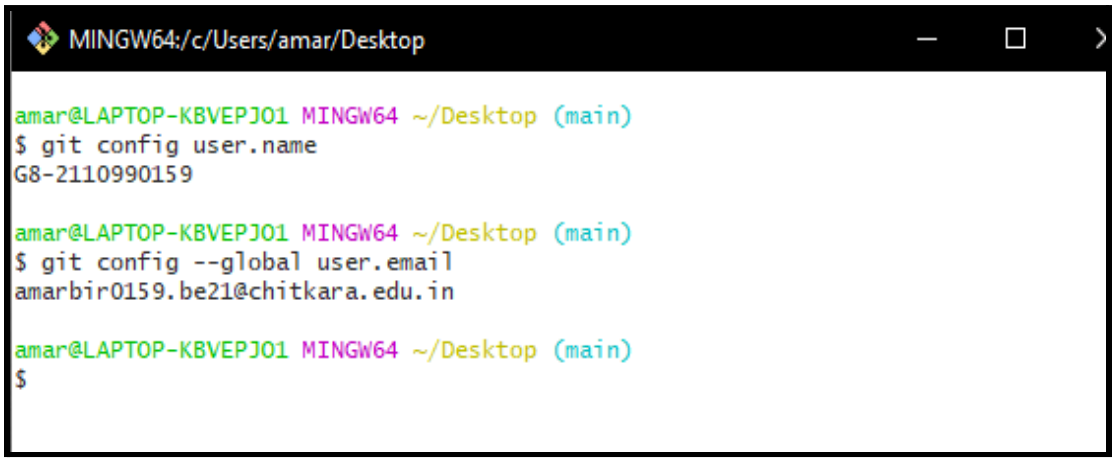
2.



3. Configuration of Git:

Step1)

You can configure your Git by typing: -
1.Set your username: git config --global user.name "Your Name"
2.Set your email address: git config --global user.email "Your Email
The page looks like as: -



Step2)

You can check configuration of Git by typing -
1.git config --list
The page looks like as: -

Program to Generate logs:

Advantage of version control systems like git is that it can record changes. 'Git log' command is used to display all these changes that were made by the user in the repository. Log is a record of all the previous commits.

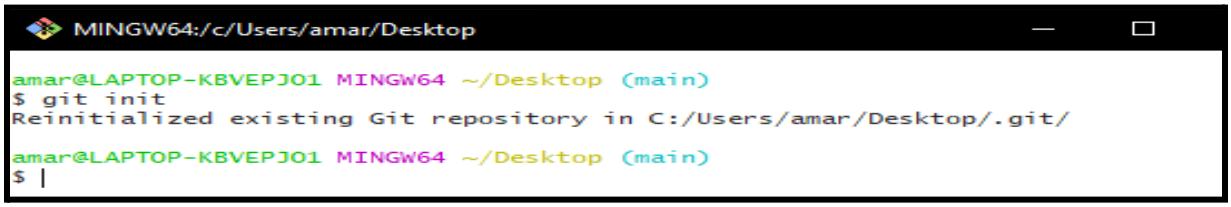
To understand Logs we need to get familiar with all the commands that are used in making changes to a repository.

1)Repository: A repository is a directory that contains all the project-related data.
2)Git init: The git init command is used to create a new blank repository.
3)Git status: We can list all the untracked, modified and deleted files using the git status command.
4)Git add: Adds all the untracked and modified files to the staging area.
5)Git commit: Git commit finalizes the changes in the repository. Every commit is saved in the branch you are working in and can be used to revert back to older versions of the project.

Making GIT Repository

Step1: Git init

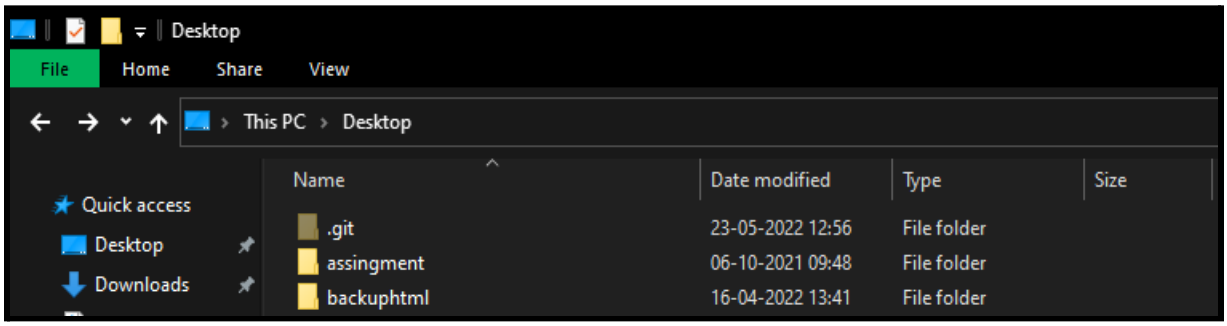
Initializing a new repository, You Can do it by typing: -
1.git init
The page looks like as: -



Step2: Adding the files to the folder

Just like (Samples...)

The page looks like as: -



Step3: Git add

The git add command adds a change in the working directory to the staging area.

You Can do it by typing: -

1.git add
2.git add (current file name)

The page looks like as: -

```
amar@LAPTOP-KBVEPJO1 MINGW64 /d/g1 (master)
$ git add .

amar@LAPTOP-KBVEPJO1 MINGW64 /d/g1 (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   first.cpp
```

Step4: Git Commit

The "commit" command is used to save your changes to the local repository.
You Can do it by typing: -
1.git commit -m”any text”

The page looks like as: -

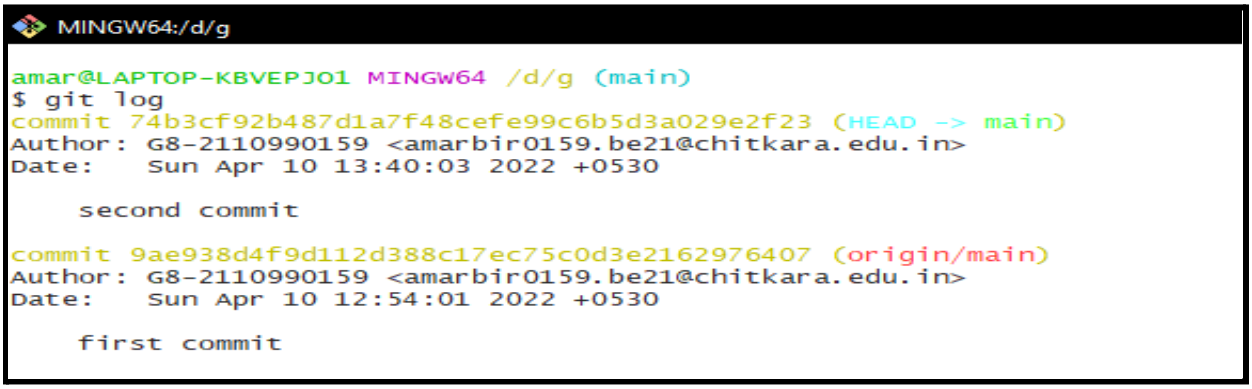
```
amar@LAPTOP-KBVEPJO1 MINGW64 /d/g1 (master)
$ git commit -m "added headings to first.cpp"
[master 6df3564] added headings to first.cpp
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Step5: Git Log

Git log will show all the commits made by the author with their time.
After every commit the checksum value (written In yellow color) of the folder changes.

Checksum is used to verify that the data in that file has not been tampered with or manipulated, possibly by a malicious entity.

You Can do it by typing: -
1.git log
The page looks like as: -



Step6: Create and Visualize Branches

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master.

Step1: CHECKING UP THE BRANCHES

You can check which branch you are working in by using the command

1.'git branch'.

The default branch is always the master branch.

The page looks like as: -

```
amar@LAPTOP-KBVEPJO1 MINGW64 /d/g (main)
$ git branch
  feature
* main

amar@LAPTOP-KBVEPJO1 MINGW64 /d/g (main)
$ git branch feature
fatal: a branch named 'feature' already exists
```

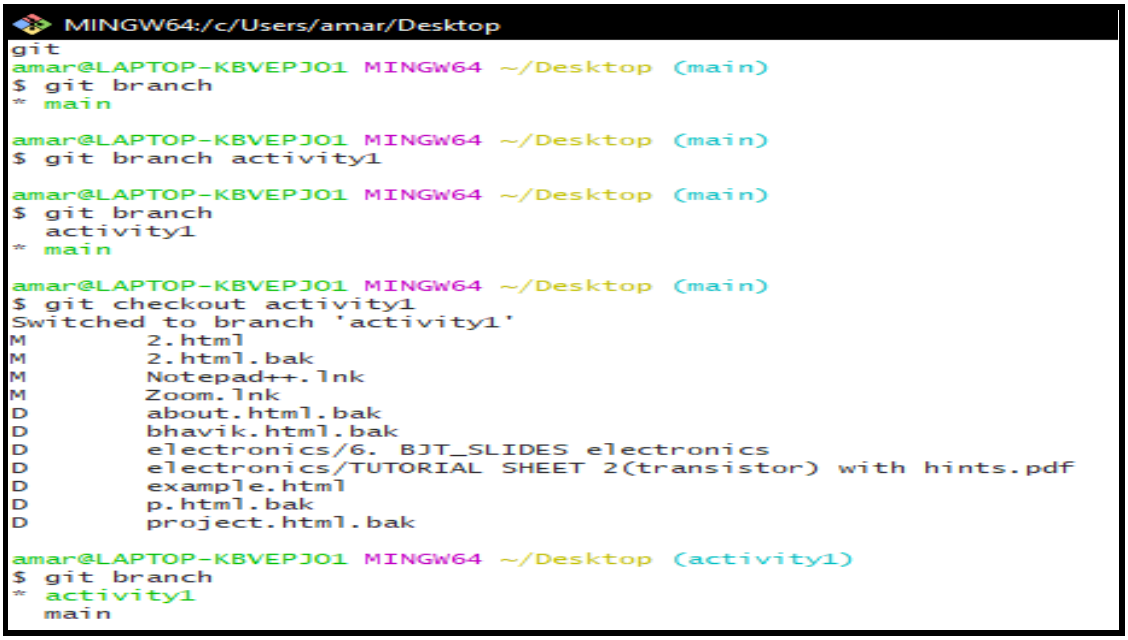
Step3: Changing Branches

To switch to the other branch

You Can do it by typing: -

1.git checkout (BRANCH NAME)

The page looks like as: -

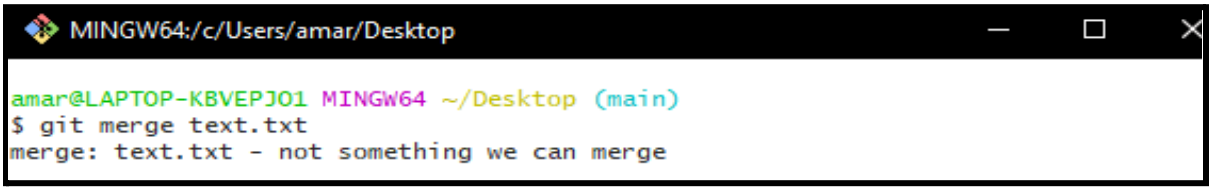


Step6: GIT MERGING

Now you can merge two branches by command.
1.git merge (BRANCH NAME)

If you want to merge a new branch in master branch you need to first checkout into the master branch and then run the command.

The page looks like as: -



There are three stages for git lifecycle:
1)Working directory
2)Staging area
3)Git repository
Working Directory:

The working directory is the folder in your local computer where the project files and folders are stored.

The local directory is created by the command ‘git init’ which creates a ‘.git’ named folder which is used to track the files in the directory.
‘.git folder’ is generally hidden but can be tracked enabling hidden files.

Staging area:

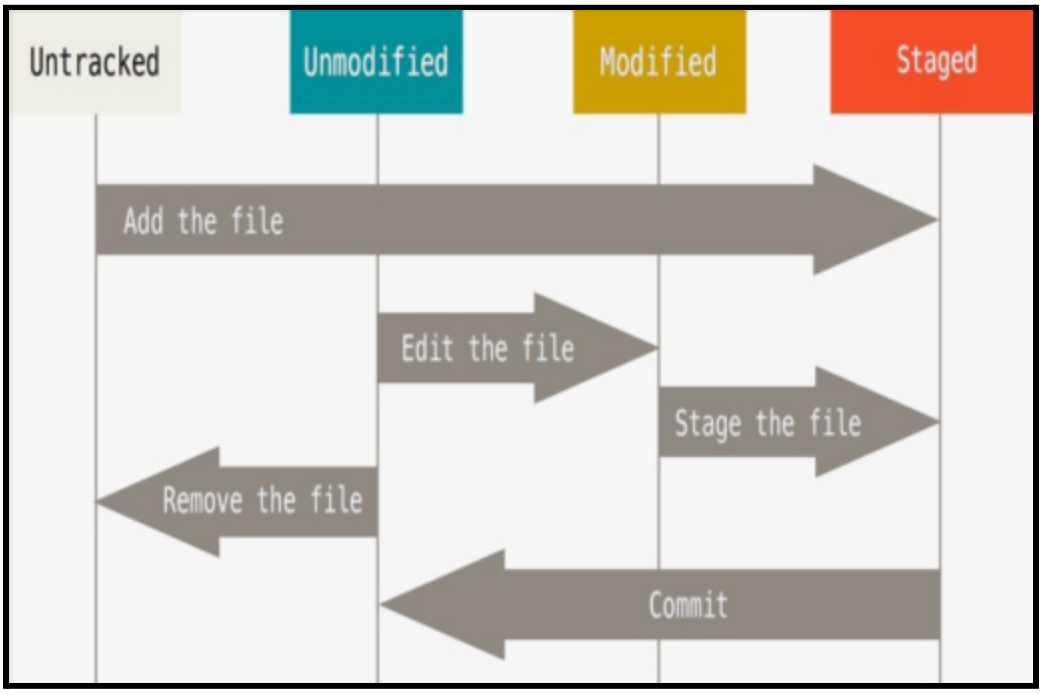
The staging area has those files which are supposed to go to the next commit. Only those files which are needed to go to the next commit stay in the staging area.

You can shift the files to the git repository by using the command
‘git add --a’.

Git repository:

Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the git commit command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about.

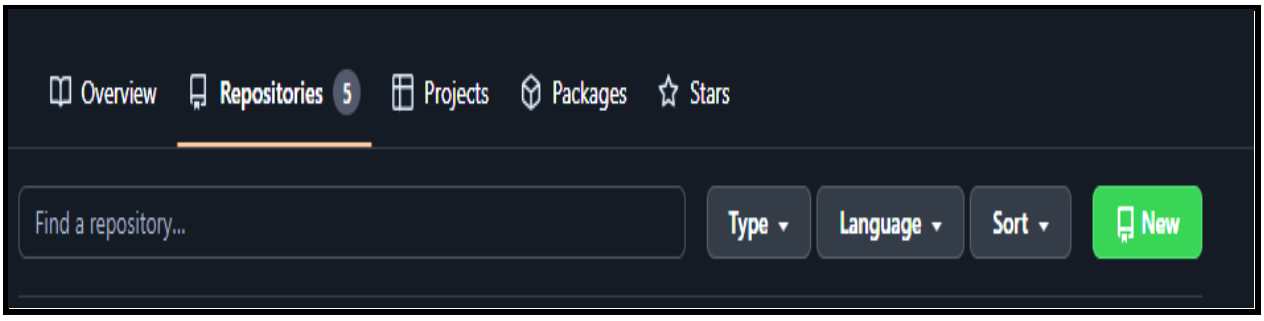
You can commit files by using command ‘git commit -m “message”’



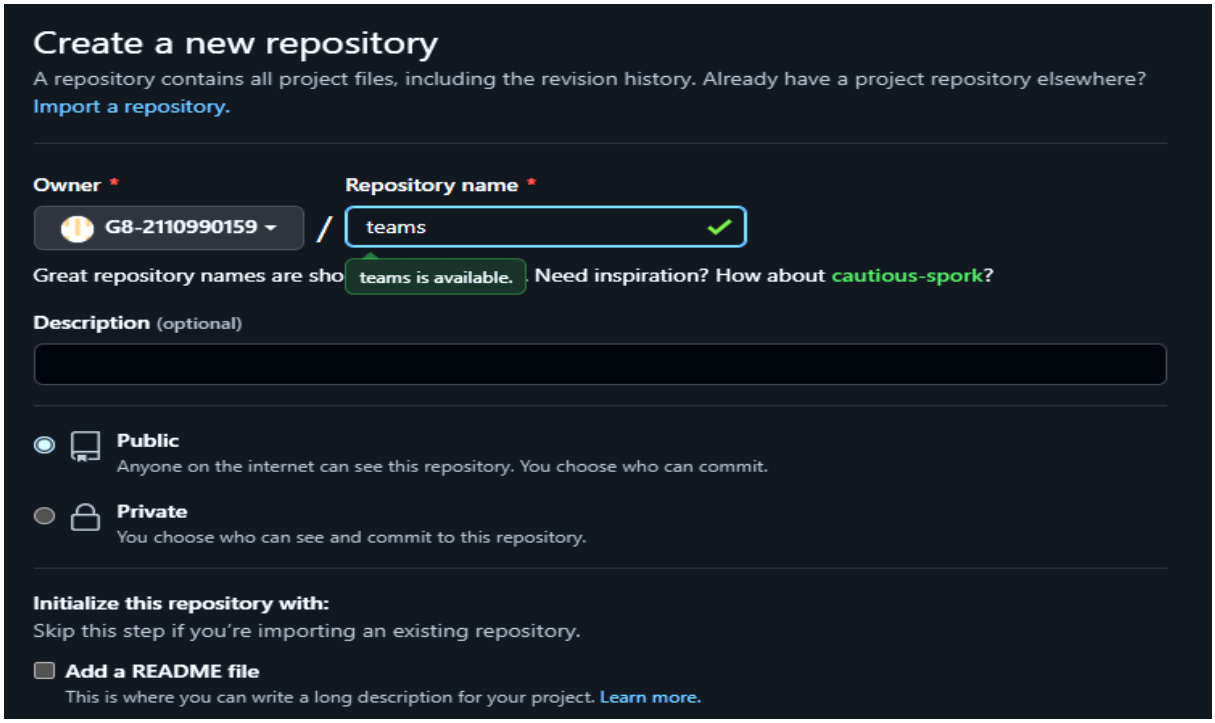
Uploading data on github

NOTE-
YOU HAVE TO MAKE A REPOSITORY IN GITHUB.
Step1) CREATING REPOSITORY IN GITHUB

The page looks like as: -



By clicking on new you are able to make a new repository.



Write the repository name and click on next.

Your GITHUB Repository has been created.

Step2) GIT ADDING REMOTE BRANCH

Git stores a branch as a reference to a commit, and a branch represents the tip of a series of commits.

You Can do it by typing: -
1.git branch -M main

The page looks like as: -

```
amar@LAPTOP-KBVEPJO1 MINGW64 /d/g (master)
$ git branch -M main

amar@LAPTOP-KBVEPJO1 MINGW64 /d/g (main)
```

Step3) GIT ADDING REMOTE ORIGIN

Is a Git repository that's hosted on the Internet
You Can do it by typing: -
1.git remote add origin

The page looks like as: -

```
amar@LAPTOP-KBVEPJO1 MINGW64 /d/g (master)
$ git remote add origin https://github.com/Group08-Chitkara-University
/2110990159.git
```

Step4) GIT CLONE

The git clone command is used to upload local repository content to a remote repository.

You Can do it by typing: -
1.git clone url

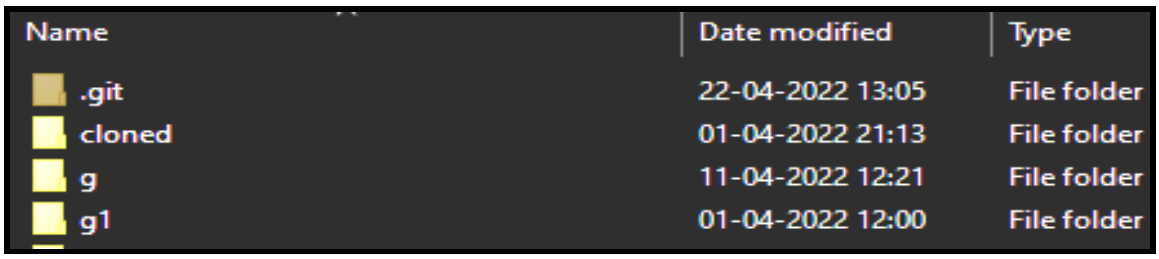
The page looks like as: -

```
amar@LAPTOP-KBVEPJO1 MINGW64 /d/cloned
$ git clone https://github.com/G8-2110990159/g1
Cloning into 'g1'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 12 (delta 2), reused 12 (delta 2), pack-reused 0
Receiving objects: 100% (12/12), done.
Resolving deltas: 100% (2/2), done.
```

Final Result

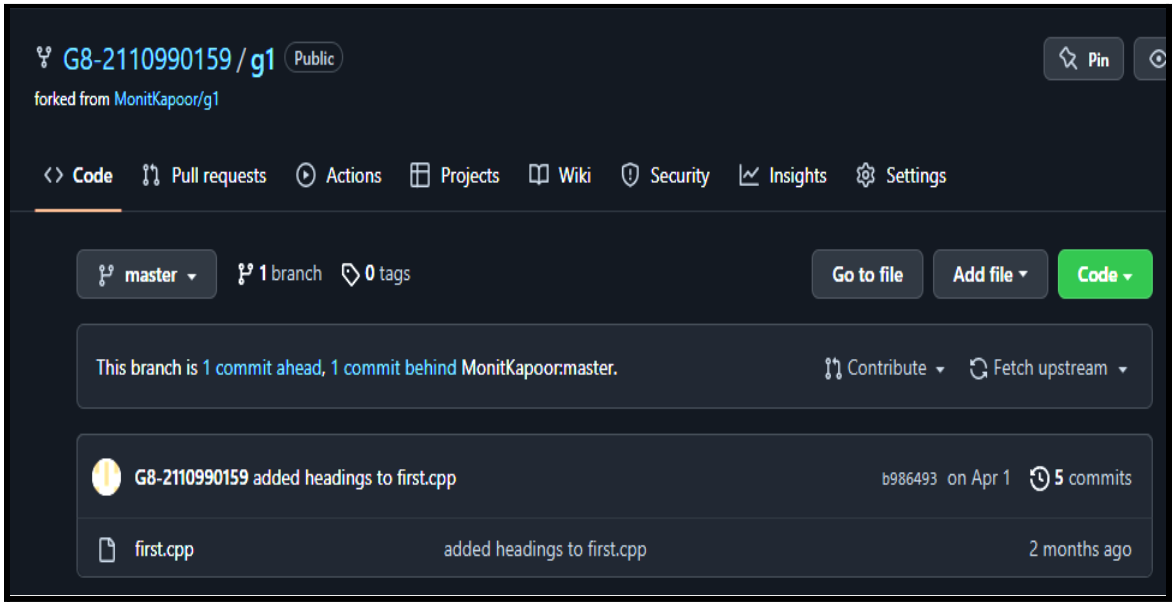
1.Document in your system

The page looks like as: -



2. Document in your GITHUB Repository

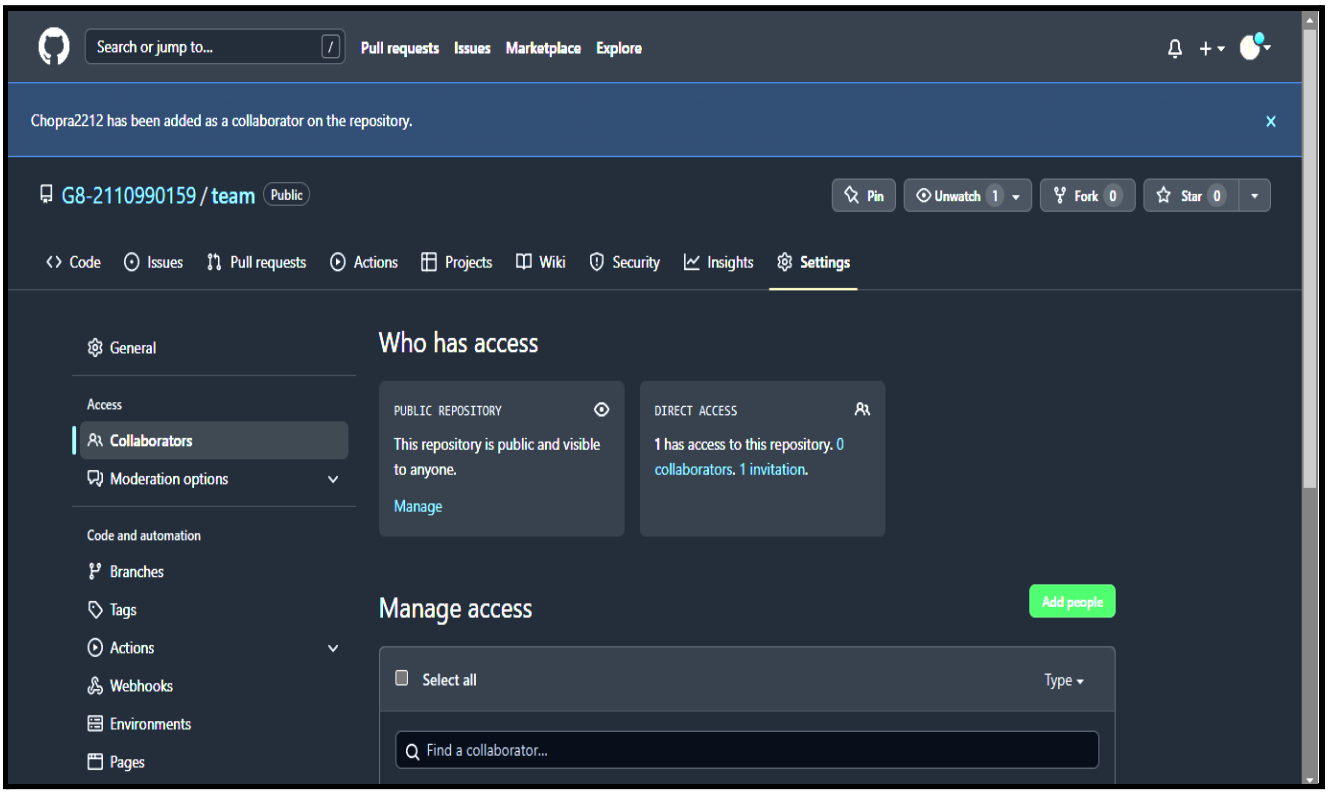
The page looks like as: -

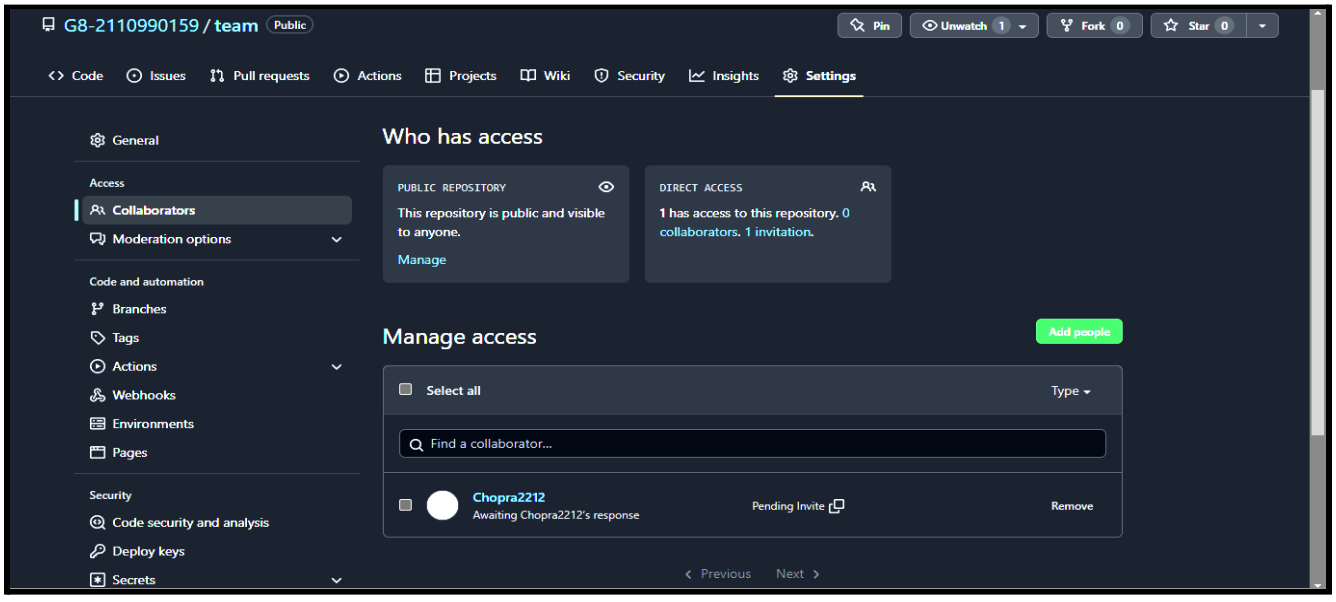


Task 1.2

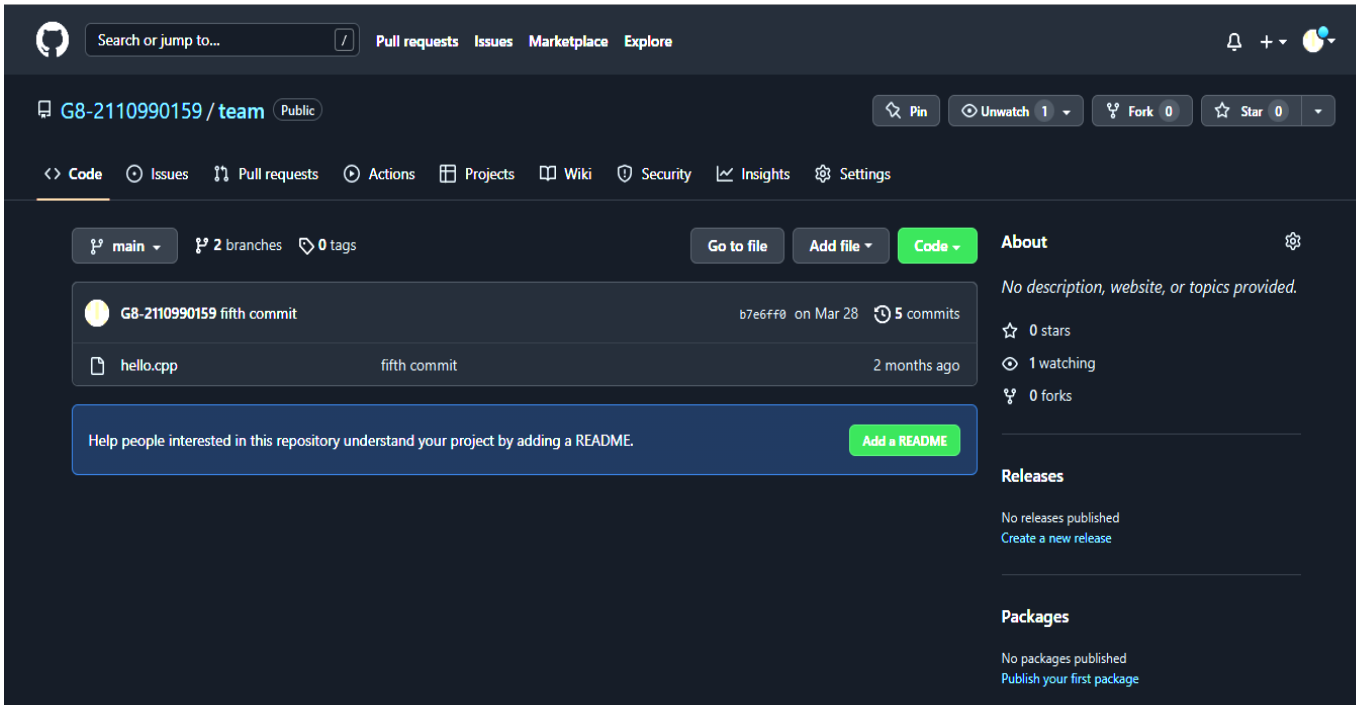
1. Add Collaborators on Github Repo :-

1. Ask for the username of the person you're inviting as a collaborator. ...

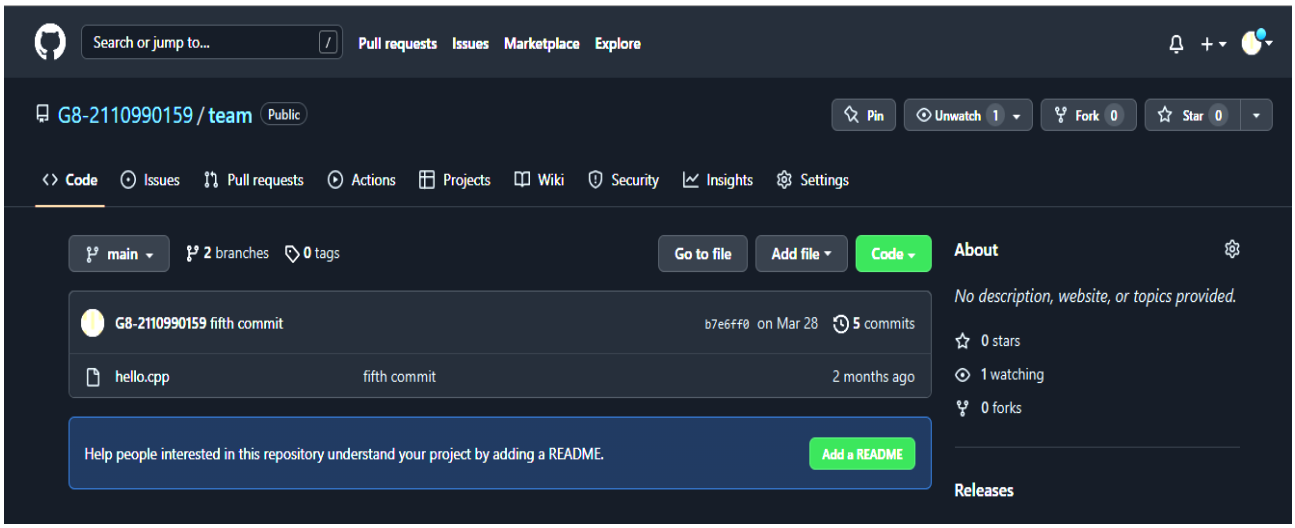




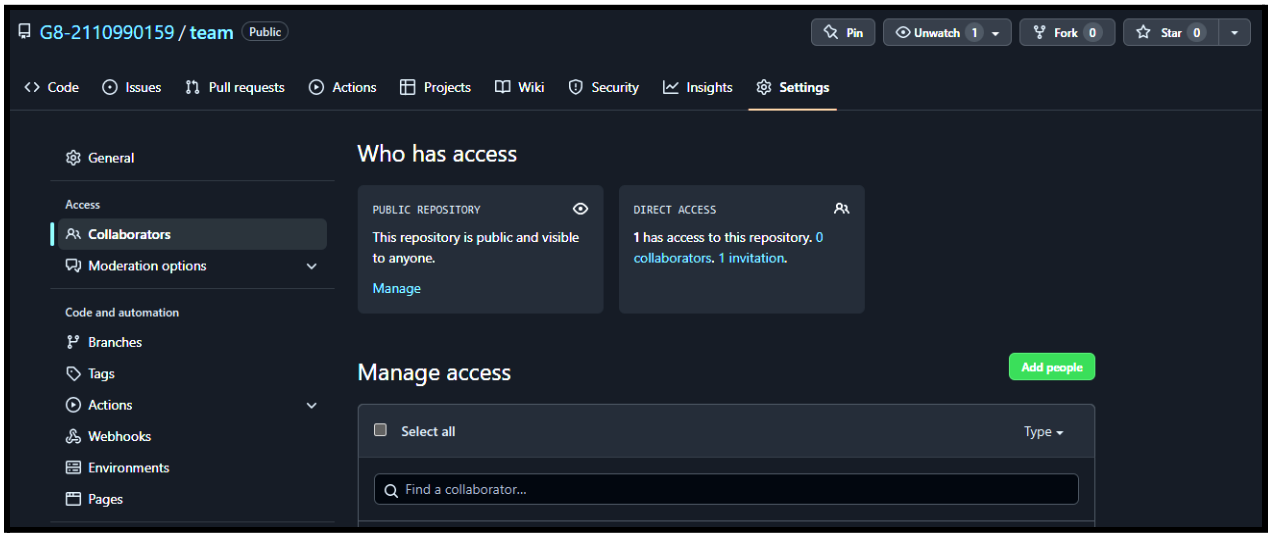
2. On GitHub.com, navigate to the main page of the repository.



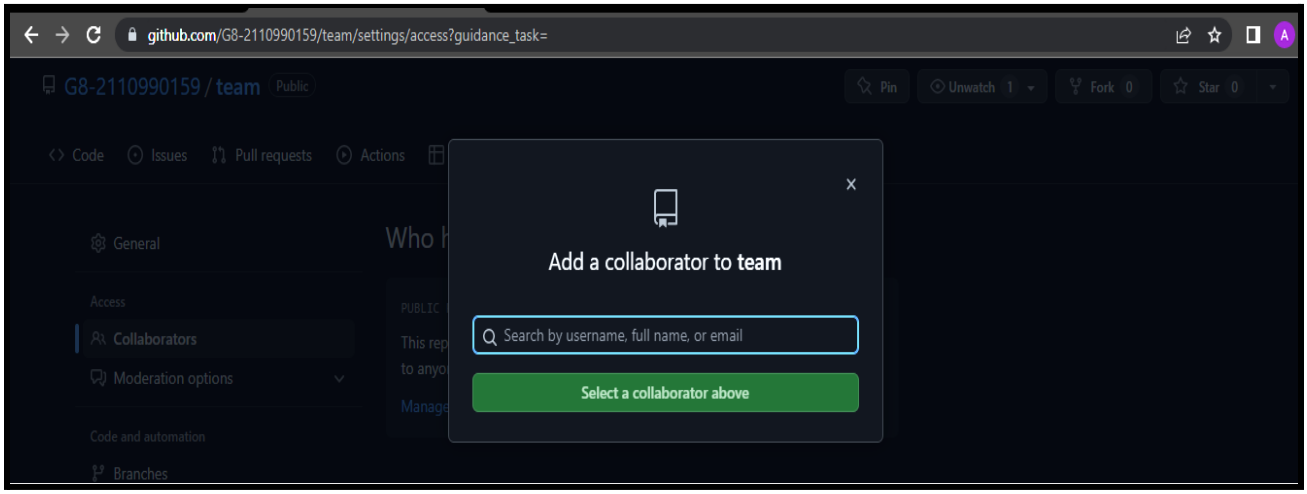
3. Under our repository name, click Settings

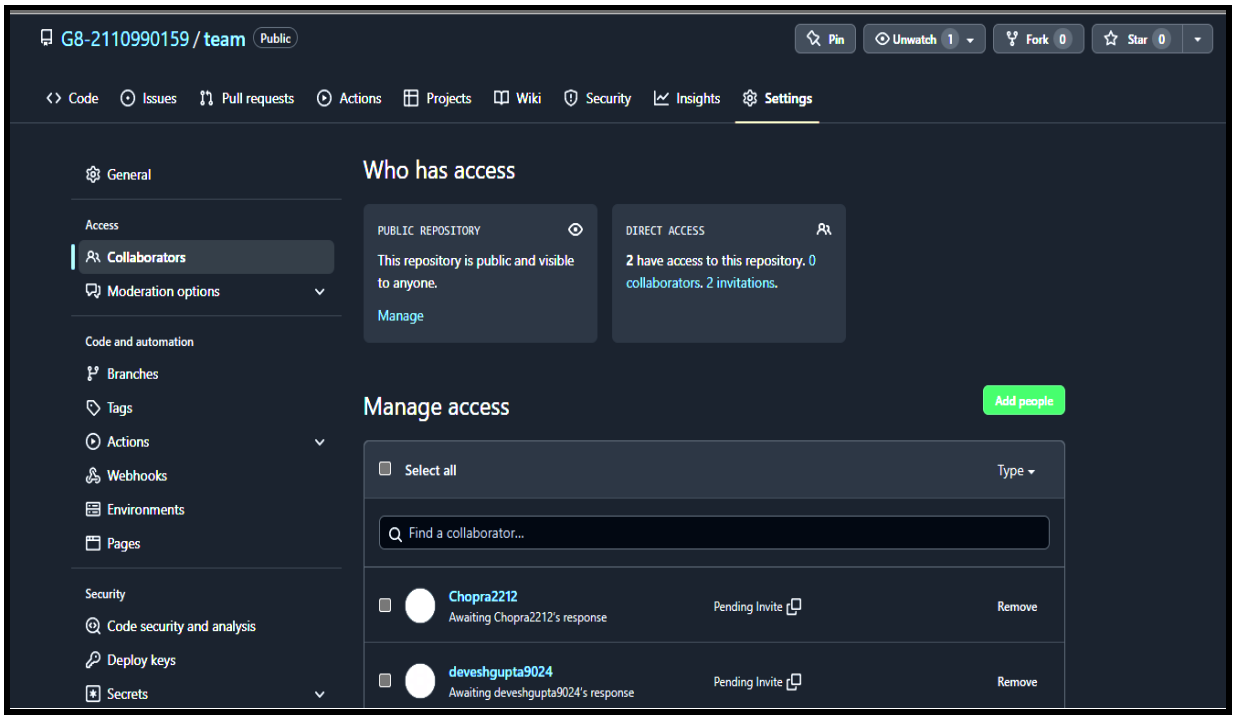


4. In the "Access" section of the sidebar, click Collaborators & teams.



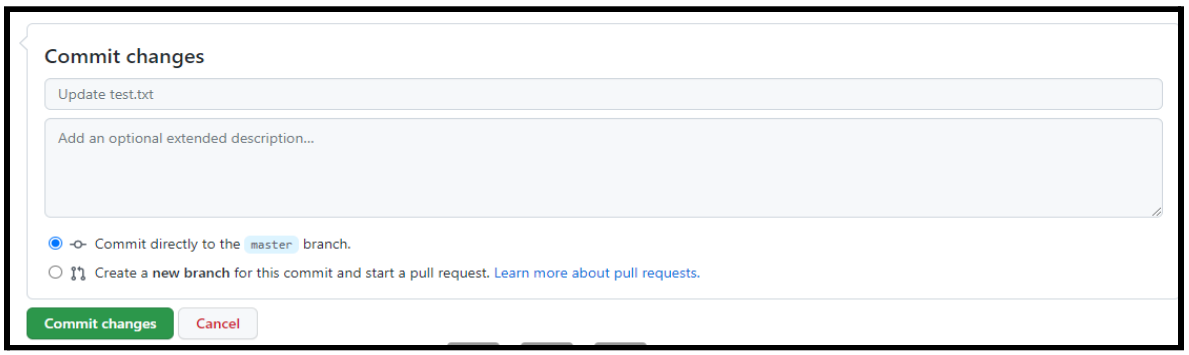
5. Click Invite a collaborator.

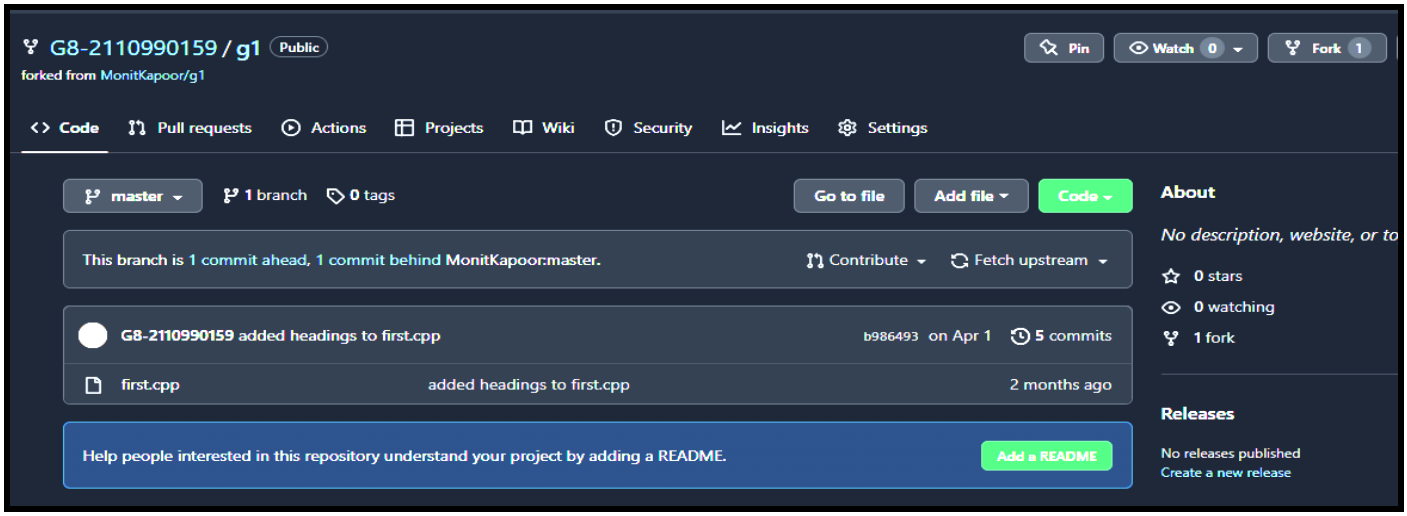




2. Fork And Commit :- A fork is a copy of a repository that we manage. Forks let us make changes to a project without affecting the original repository. We can fetch updates from or submit changes to the original repository with pull requests.

If we need to fork a GitHub or GitLab repo, it's as simple as navigating to the landing page of the repository in your web browser and clicking on the Fork button on the repository's home page. A forked copy of that Git repository will be added to your personal GitHub or GitLab repo. That's it. That's all we have to do to fork a Git repo.





3. Merge and Resolve conflicts created due to own Activity and Collaborators activity:-

There are a few steps that could reduce the steps needed to resolve merge conflicts in Git.

1. The easiest way to resolve a conflicted file is to open it and make any necessary changes.

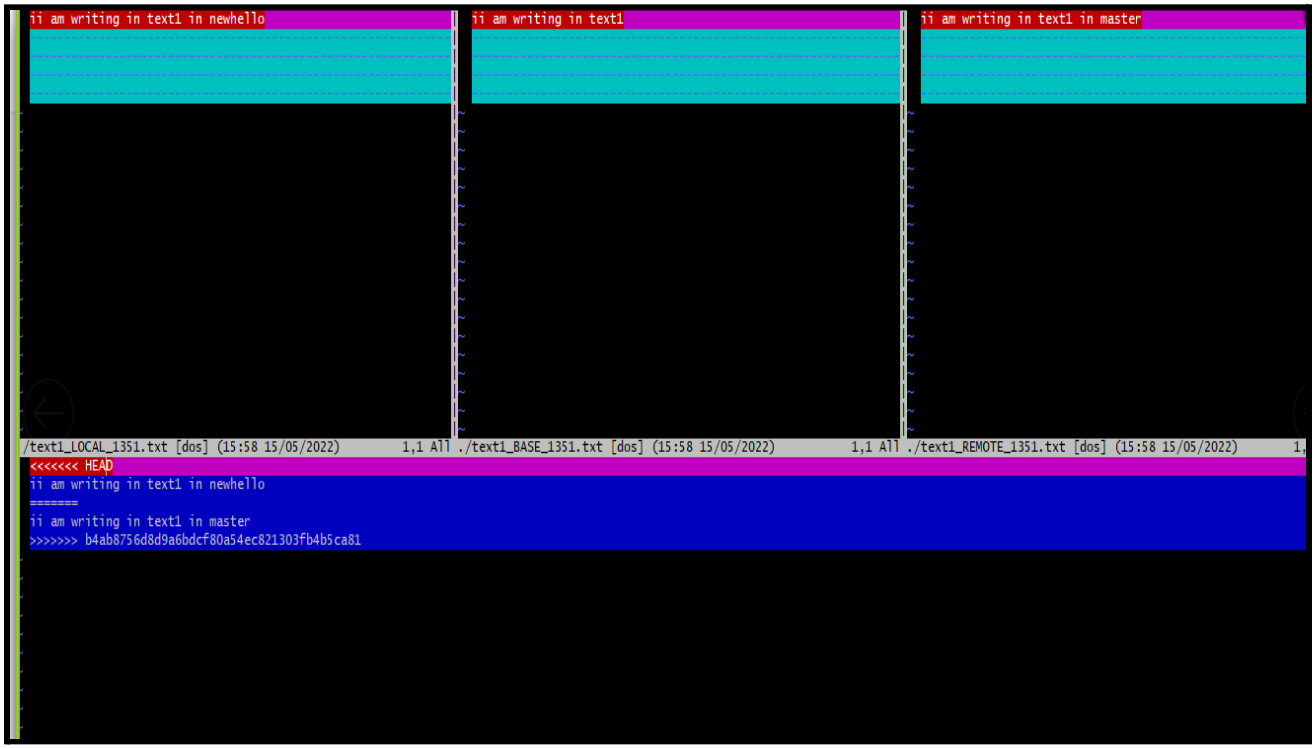
2. After editing the file, we can use the git add a command to stage the new merged content.

3. The final step is to create a new commit with the help of the git commit command.

4. Git will create a new merge commit to finalize the merge

Let us now look into the Git commands that may play a significant role in resolving conflicts.

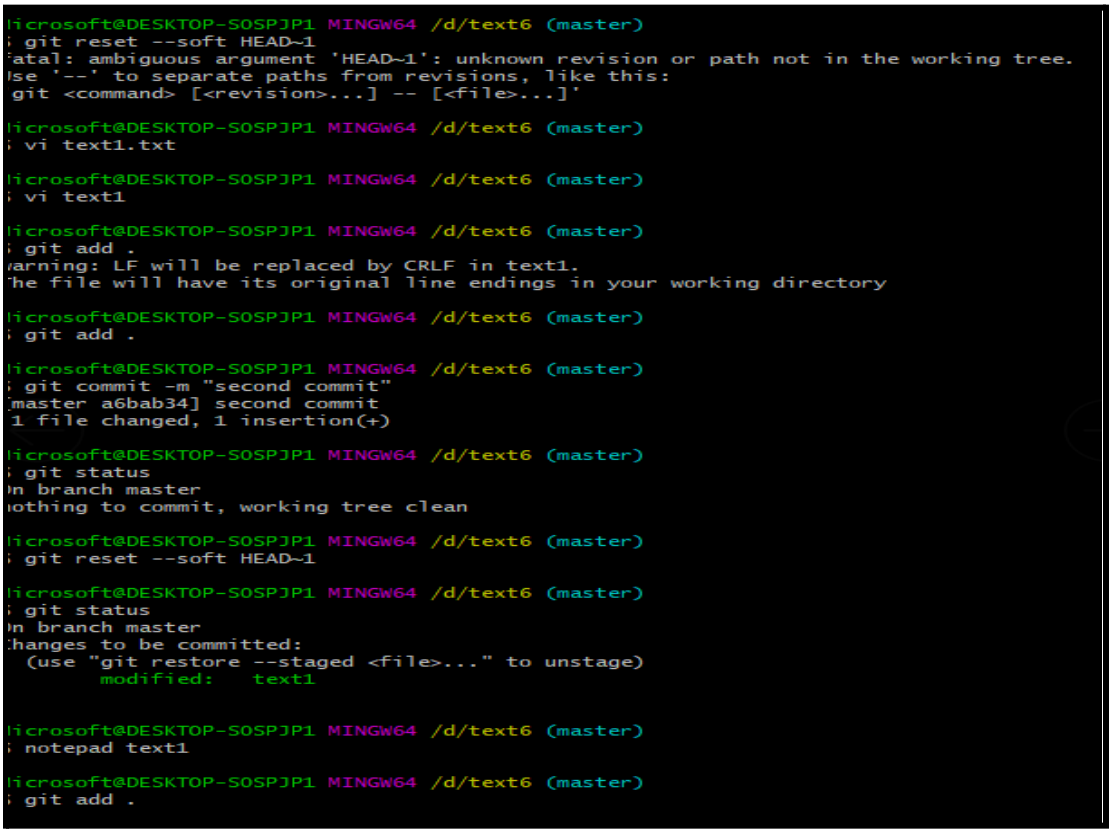
```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello|MERGING)
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge ecmerge p4merge araxis bc codecompare smerge emerge vimdiff nvimdiff
Merging:
text1.txt

Normal merge conflict for 'text1.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
4 files to edit

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello|MERGING)
$ git status
On branch newhello
Your branch is up to date with 'origin/newhello'.

All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        text1.txt.orig


Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/hello (newhello|MERGING)
$ git add .
```



4. Git Reset :- Git reset is a powerful command that is used to undo local changes to the state of a Git repo. Git reset operates on "The Three Trees of Git". These trees are the Commit History (HEAD), the Staging Index, and the Working Directory.

The easiest way to undo the last Git commit is to execute the “git reset” command with the “–soft” option that will preserve changes done to your files.

Git reset --hard , which will completely destroy any changes and remove them from the local directory.



```
Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git commit -m "second commit"
[master fc546b9] second commit
 1 file changed, 1 insertion(+)

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git reset --hard HEAD~1
HEAD is now at 7487caf first commit

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ notepad text1

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ touch text2.bin

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ touch .gitignore

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ vi .gitignore

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ vi text2.bin

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ cat text2.bin
hello i am writing in text2

Microsoft@DESKTOP-SOSPJP1 MINGW64 /d/text6 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```