.

Subject Name: Source Code Management
Subject Code: CS181
Cluster: Beta
Department: DCSE

Submitted To: Mr Monit Kapoor

Submitted By: Aastik Shukla

Roll no.21109900338

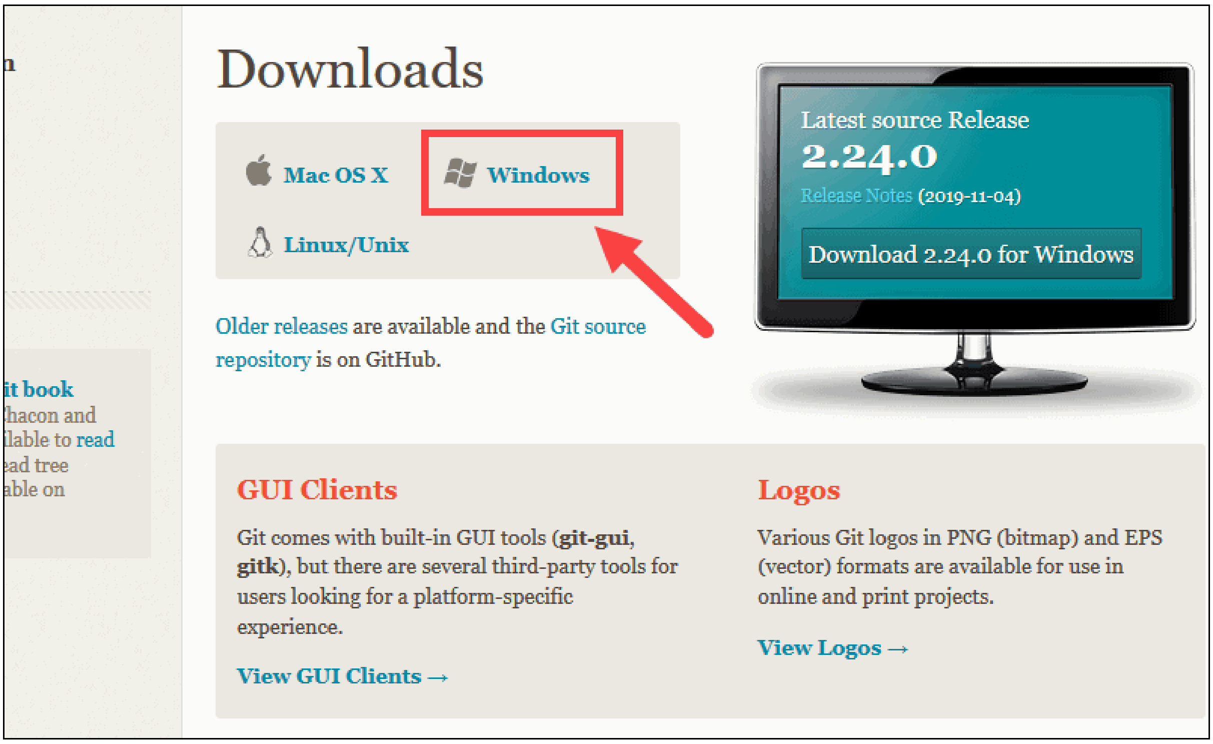
G8-A

Lists Of Tasks

Task – 1
Aim: Setting up git client.

Procedure to install git for Windows:

1. **Browse to the official Git website: https://gitscm.com/downloads**
2. **Click the download link for Windows and allow the download to complete.**



Downloads

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

Latest source Release
2.24.0
Release Notes (2019-11-04)

it book
hacon and
ilable to read
ad tree
able on

GUI Clients

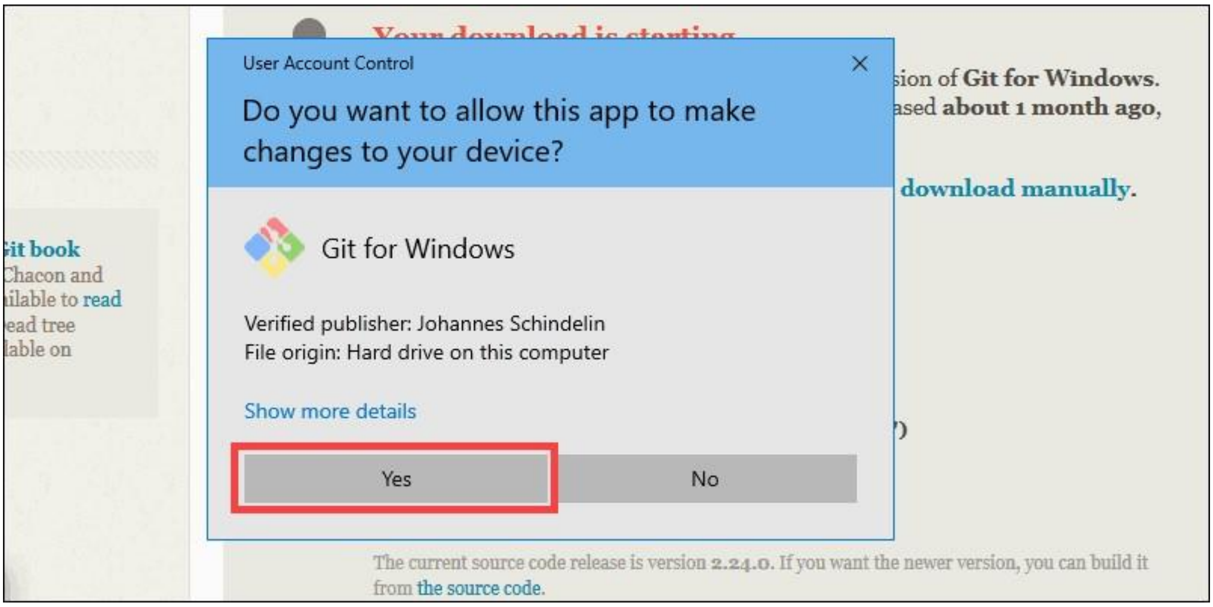
Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

Logos

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

3. **Browse to the download location (or use the download shortcut in your browser). Double-click the file to extract and launch the installer.**

4. **Allow the app to make changes to your device by clicking Yes on the User Account Control dialog that opens.**



User Account Control

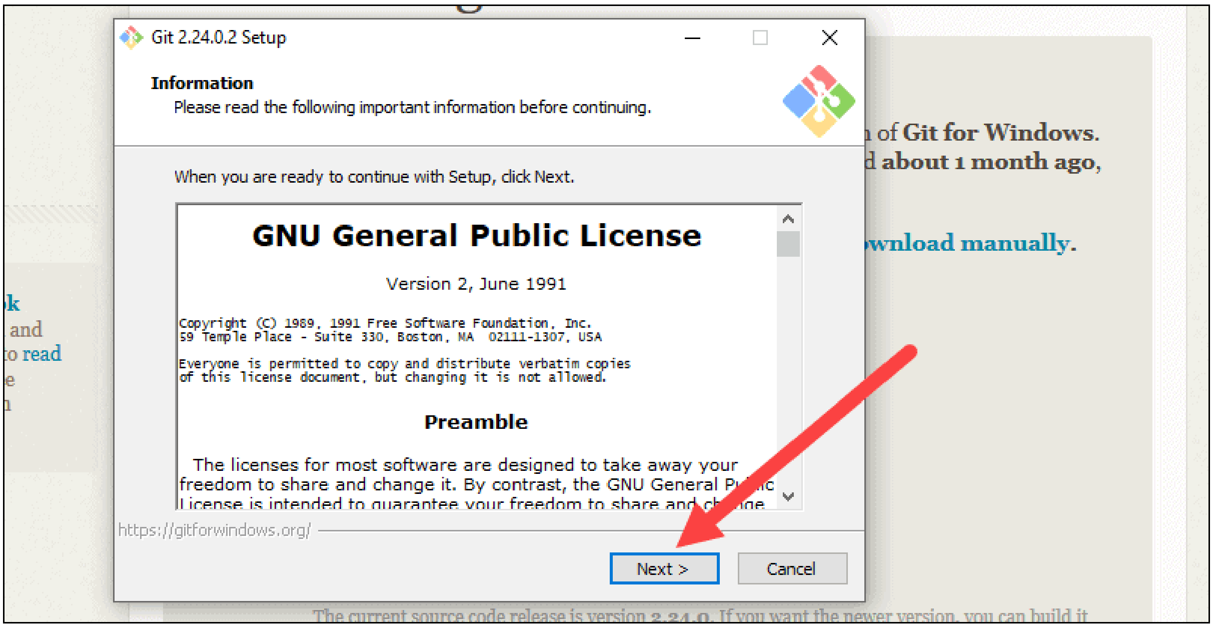
Do you want to allow this app to make changes to your device?

Git for Windows

Verified publisher: Johannes Schindelin
File origin: Hard drive on this computer

The current source code release is version 2.24.0. If you want the newer version, you can build it from the source code.

5. **Read the GNU General Public License, and when you're ready to install, click Next.**



Git 2.24.0.2 Setup

Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

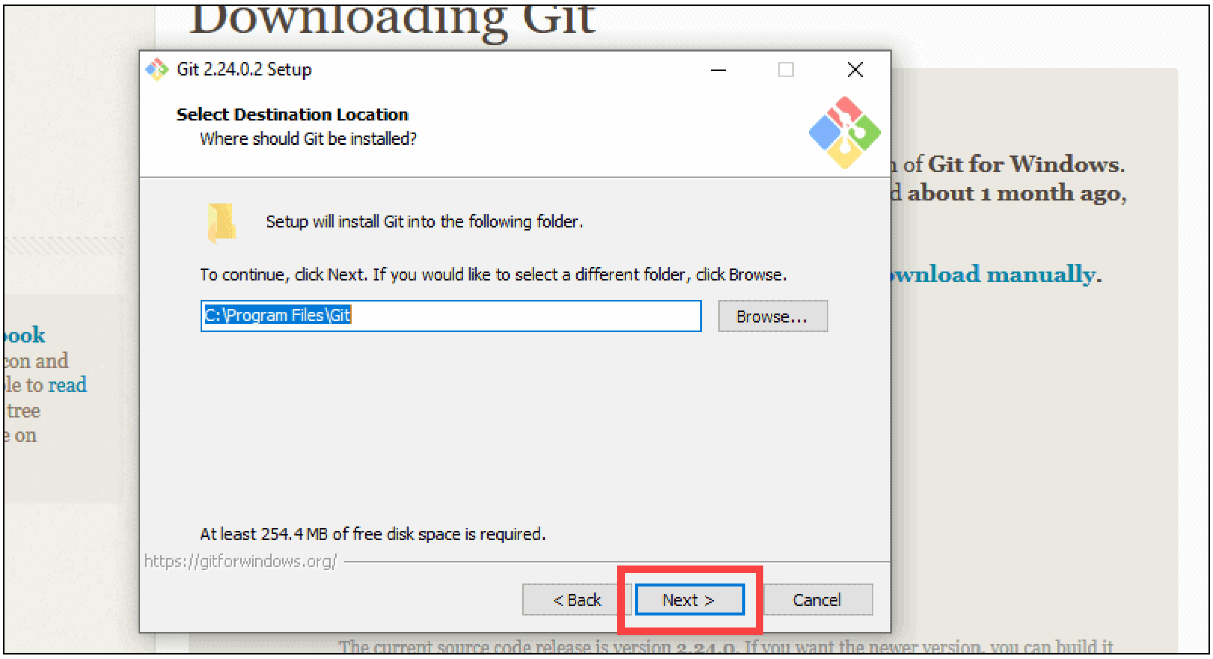
Version 2, June 1991

```
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA  02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

6. **The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click Next.**



Git 2.24.0.2 Setup

Select Destination Location

Where should Git be installed?

Setup will install Git into the following folder.

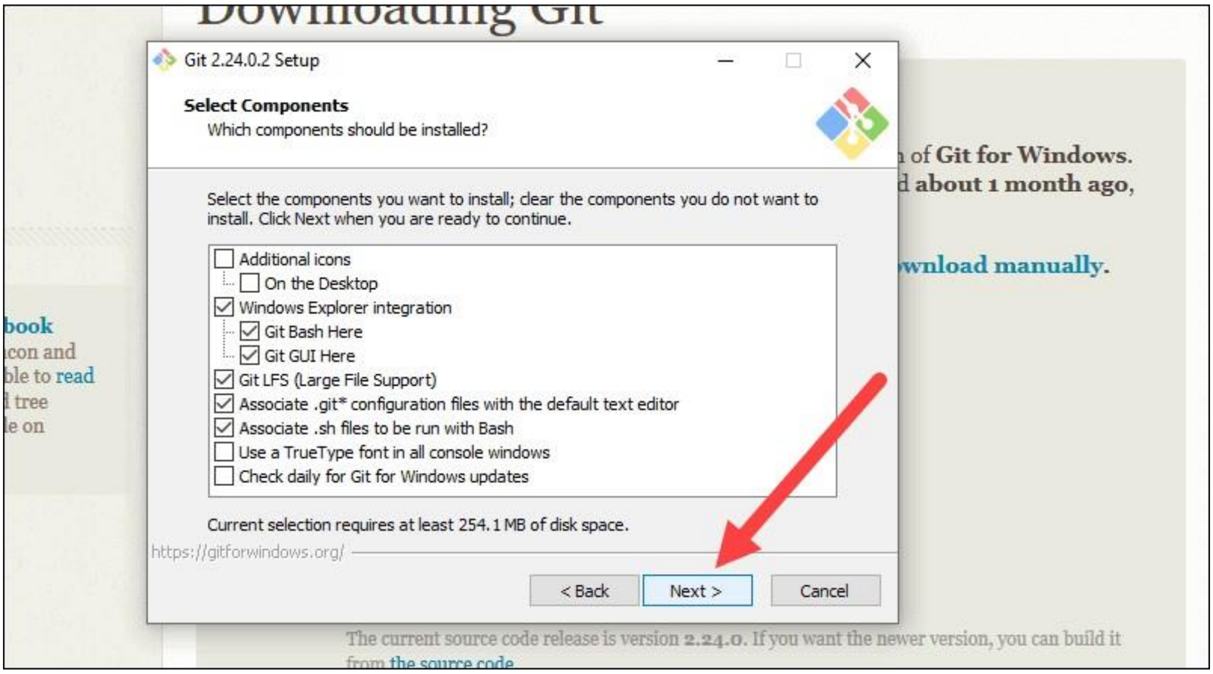
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Git

At least 254.4 MB of free disk space is required.

https://gitforwindows.org/

7. **A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click Next.**



Git 2.24.0.2 Setup

Select Components

Which components should be installed?

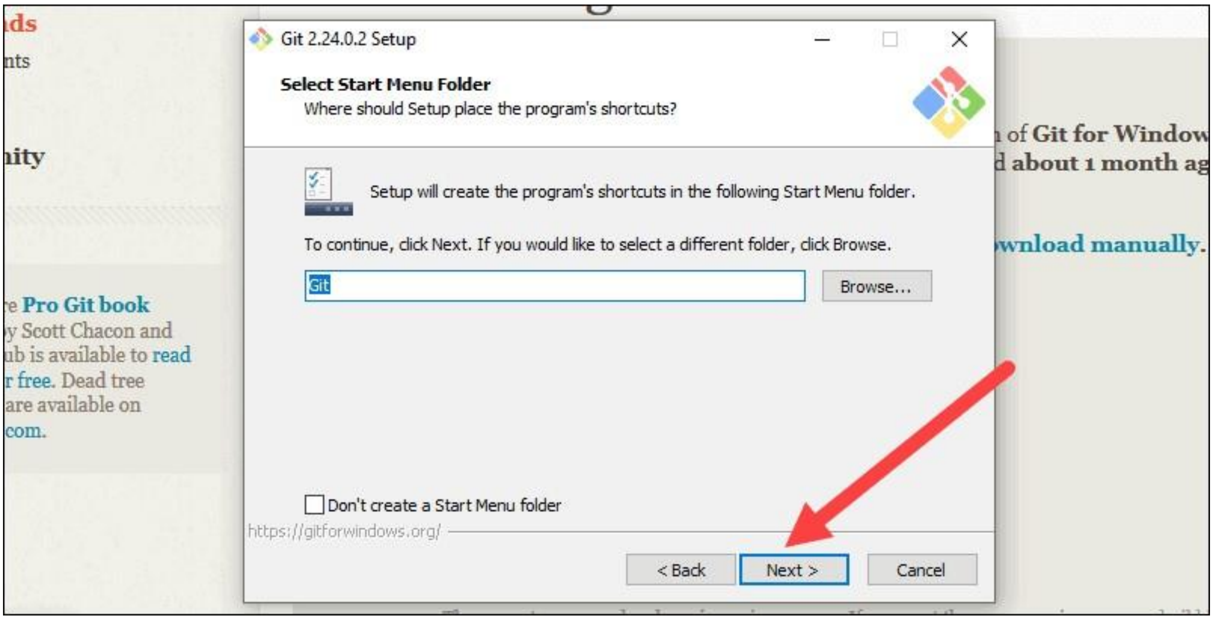
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- □ Additional icons
  - □ On the Desktop
- ☑ Windows Explorer integration
  - ☑ Git Bash Here
  - ☑ Git GUI Here
- ☑ Git LFS (Large File Support)
- ☑ Associate .git* configuration files with the default text editor
- ☑ Associate .sh files to be run with Bash
- □ Use a TrueType font in all console windows
- □ Check daily for Git for Windows updates

Current selection requires at least 254.1 MB of disk space.

https://gitforwindows.org/

8. **The installer will offer to create a start menu folder. Simply click Next.**



Git 2.24.0.2 Setup

Select Start Menu Folder

Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

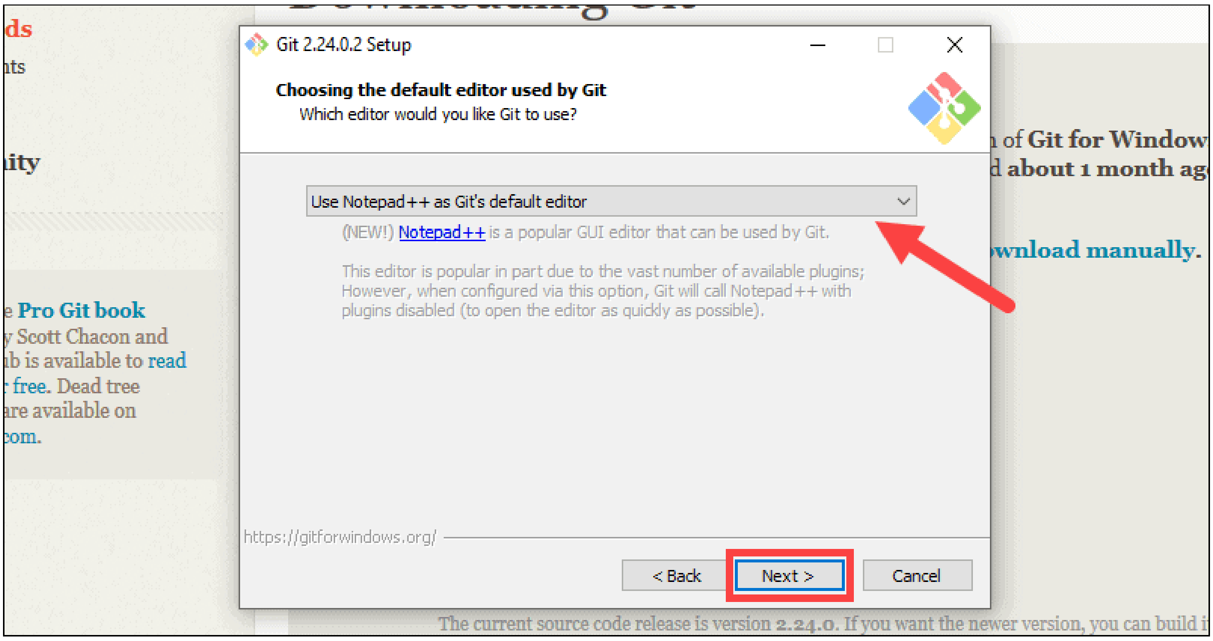
To continue, click Next. If you would like to select a different folder, click Browse.

Git

□ Don't create a Start Menu folder

https://gitforwindows.org/

9. **Select a text editor you'd like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click Next.**



Git 2.24.0.2 Setup

Choosing the default editor used by Git

Which editor would you like Git to use?

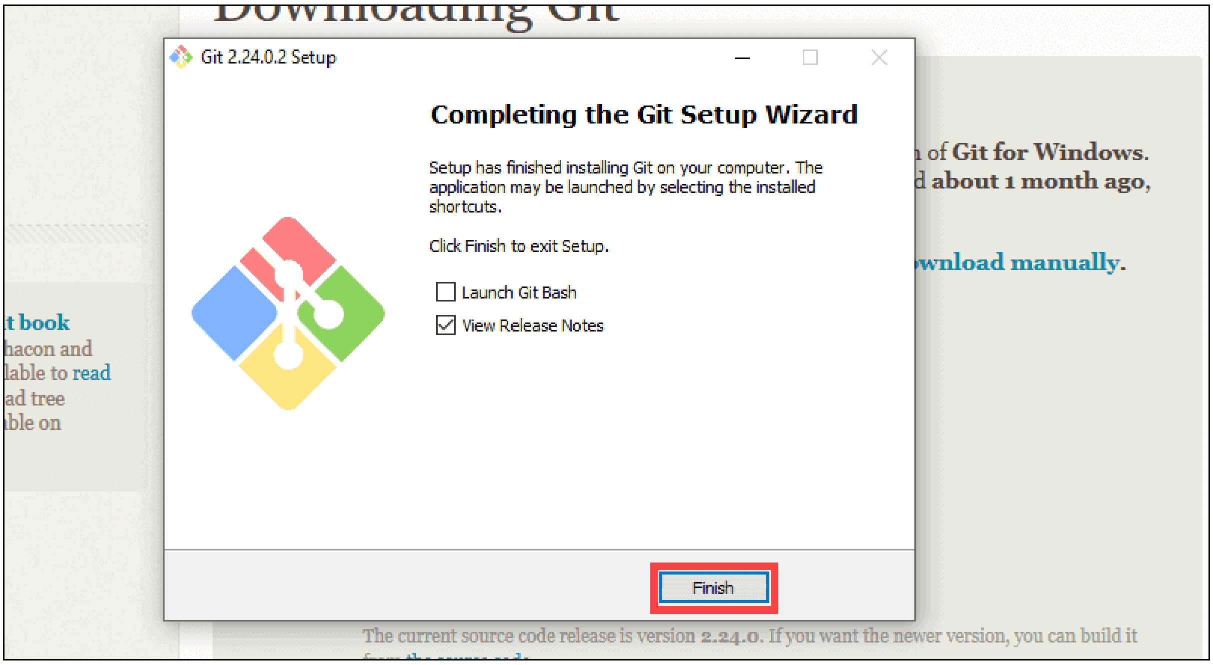
Use Notepad++ as Git's default editor

(NEW!) Notepad++ is a popular GUI editor that can be used by Git.

This editor is popular in part due to the vast number of available plugins; However, when configured via this option, Git will call Notepad++ with plugins disabled (to open the editor as quickly as possible).

https://gitforwindows.org/

10. **Once the installation is complete, tick the boxes to view the Release Notes or Launch Git Bash, then click Finish.**



Completing the Git Setup Wizard

Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

□ Launch Git Bash

☑ View Release Notes

Configuring Git to ignore certain files:

This part is extra important and required so that your repository does not get cluttered with garbage files. By default, Git tracks all files in a project. Typically, this is not what you want; rather, you want Git to ignore certain files such as .bak files created by an editor or .class files created by the Java compiler. To have Git automatically ignore particular files, create a file named .gitignore (note that the filename begins with a dot) in the C:\users\name folder (where name is your MSOE login name).

NOTE: The. gitignore file must NOT have any file extension (e.g. .txt). Windows normally tries to place a file extension (.txt) on a file you create from File Explorer - and then it (by default) HIDES the file extension. To avoid this, create the file from within a useful editor (e.g. Notepad++ or Ultra Edit) and save the file without a file extension)

Edit this file and add the lines below (just copy/paste them from this screen); these are patterns for files to be ignored (taken from examples https://github.com/github/gitignore.) #Lines (like this one) that begin with # are comments; all other lines are rules

# common build products to be ignored at MSOE

***.o**

***.obj**

***.class**

***.exe**

# common IDE-generated files and folders to ignore workspace.xml bin / out / .classpath # uncomment following for courses in which Eclipse .project files are not checked in # .project #ignore automatically generated files created by some common applications, operating systems

***.bak**

***.log**

***.ldb**

*** .DS Store***

*** Thumbs.d**

b

# Any files you do not want to ignore must be specified starting with! # For example, if you didn't want to ignore .classpath, you'd uncomment the following rule: #

!.classpath

Once Git is installed, there is some remaining custom configuration you must do. Follow the steps below:-

a. **From within File Explorer, right-click on any folder. A context menu appears containing the commands " Git Bash here" and "GitGUI here". These commands permit you to launch either Git client. For now, select Git Bash here.**

b. **Enter the command (replacing name as appropriate) git config -- global core.exclude file c:/users/name/. gitignore**

This tells Git to use the .gitignore file you created in step 2 NOTE: TO avoid typing errors, copy and paste the commands shown here into the Git Bash window, using the arrow keys to edit the red text to match your information.

c. **Enter the command git config --global user. Email "name@chitkara.edu.in"**

This links your Git activity to your email address. Without this, your commits will often show up as "unknown login". Replace name with your own MSOE email name.

d. **Enter the command git config --global user.name "Your Name"**

Git use this to log your activity. Replace "Your Name" by your actual first and last name.

e. **Enter the command git config --global push.default simple**

This ensures that all pushes go back to the branch from which they were pulled. Otherwise pushes will go to the master branch, forcing a merge.

Task 2:-

Aim: Setting up GitHub Account

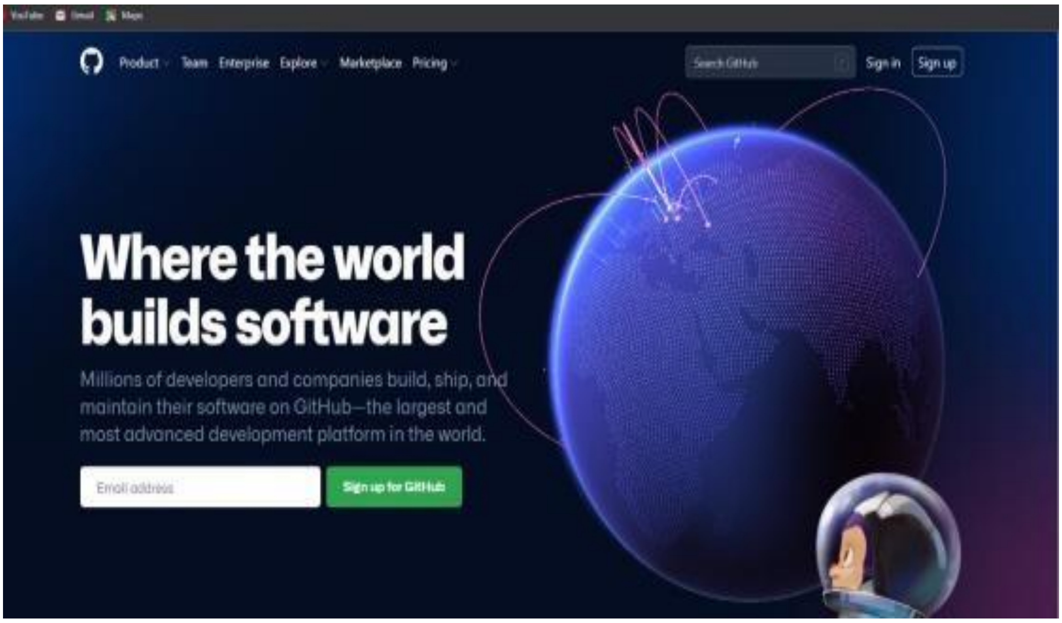
The first steps in starting with GitHub are to create an account, choose a product that fits your needs best, verify your email, set up two-factor authentication, and view your profile.

There are several types of accounts on GitHub. Every person who uses GitHub has their own user account, which can be part of multiple organizations and teams. Your user account is your identity on GitHub.com and represents you as an individual.

1. **Creating an account**

To sign up for an account on GitHub.com, navigate to https://github.com/ and follow the prompts.

To keep your GitHub account secure you should use a strong and unique password. For more information, see "Creating a strong password."



Where the world builds software

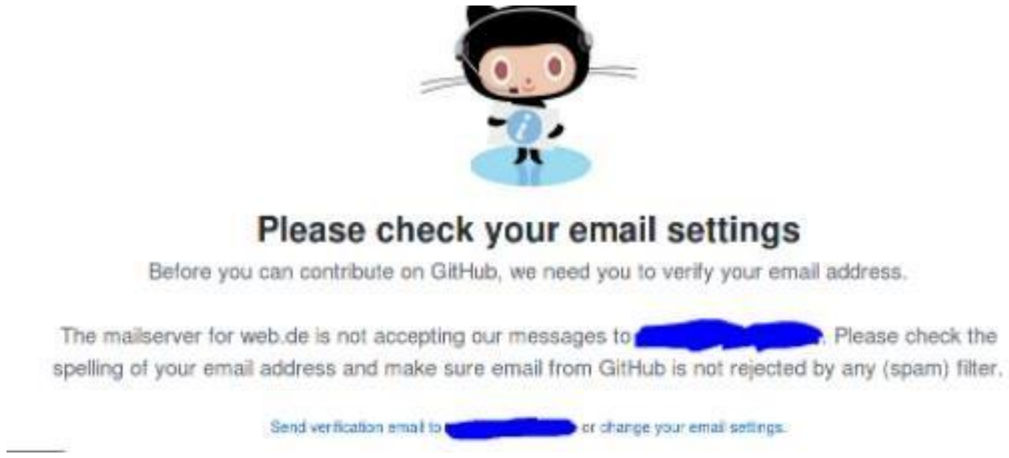
Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

2. **Choosing your GitHub product**

You can choose GitHub Free or GitHub Pro to get access to different features for your personal account. You can upgrade at any time if you are unsure at first which product you want.

For more information on all of GitHub's plans, see "GitHub's products."

3. **Verifying your email address**



Please check your email settings

Before you can contribute on GitHub, we need you to verify your email address.

The mailserver for web.de is not accepting our messages to [redacted]. Please check the spelling of your email address and make sure email from GitHub is not rejected by any (spam) filter.

Send verification email to [redacted] or change your email settings.

To ensure you can use all the features in your GitHub plan, verify your email address after signing up for a new account. For more information, see "Verifying your email address."

4. **Configuring two-factor authentication**

Two-factor authentication, or 2FA, is an extra layer of security used when logging into websites or apps. We strongly urge you to configure 2FA for the safety of your account. For more information, see "About two-factor authentication."

5. **Viewing your GitHub profile and contribution graph**

Your GitHub profile tells people the story of your work through the repositories and gists you've pinned, the organization memberships you've chosen to publicize, the contributions you've made, and the projects you've created. For more information, see "About your profile" and "Viewing contributions on your profile."

Task - 3

Aim:- Program to generate logs

Basic Git init:

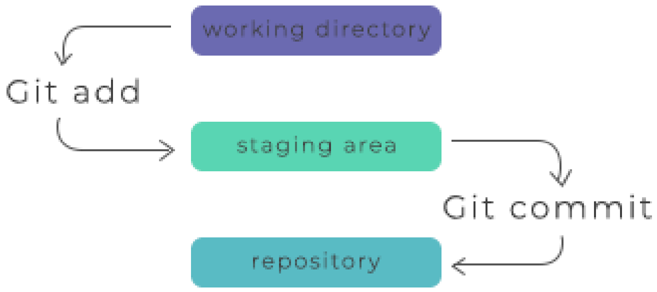
The git init is one way to start a new project with Git. To start a repository, use either git init or git clone - not both. To initialize a repository, Git creates a hidden directory called .git. That directory stores all of the objects and refs that Git uses and creates as a part of your project's history. This hidden .git directory is what separates a regular directory from a Git repository.

Basic Git Status:-

The git status command shows the state of the working directory and the staging area. It allows you to see staged changes and the files that aren't being tracked by Git. The Status output does not display any information about the committed project history. For this purpose, use the git log command. The git status simply plays what has git add and git commit been going on h commands.

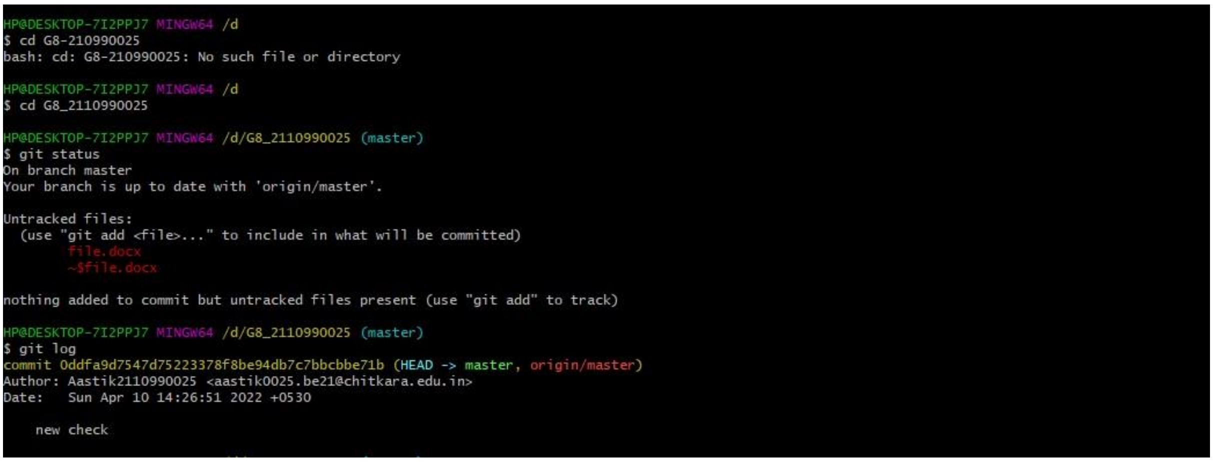
Basic Git add:-

The git add command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit. However, git add doesn't really affect the repository in any significant way— changes are not actually recorded until you run git commit



Basic Git log

Git log command is one of the most usual commands of git. It is the most useful command for Git. Every time you need to check the history, you have to use the git log command. The basic git log command will display the most recent commits and the status of the head. It will use as:



```
HP@DESKTOP-7I2PPJ7 MINGW64 /d
$ cd G8-210990025
bash: cd: G8-210990025: No such file or directory

HP@DESKTOP-7I2PPJ7 MINGW64 /d
$ cd G8_2110990025

HP@DESKTOP-7I2PPJ7 MINGW64 /d/G8_2110990025 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file.docx
        ~$file.docx

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

HP@DESKTOP-7I2PPJ7 MINGW64 /d/G8_2110990025 (master)
$ git log
commit 0ddfa9d7547d75223378f8be94db7c7bbcbbe71b (HEAD -> master, origin/master)
Author: Aastik2110990025 <aastik0025.be21@chitkara.edu.in>
Date:   Sun Apr 10 14:26:51 2022 +0530

    new check
```

Task-4
Aim:- Create and visualize branches in Git

How to Create Branches:

Git makes creating and managing branches very easy. In fact, the power and flexibility of its branching model is one of the biggest advantages of Git!

There are a couple of different use cases when creating branches in Git. Let's look at each of them in turn.

When you do a pull request on a branch, you can continue to work on another branch and make another pull request on this other branch.

Before creating a new branch, pull the changes from upstream. Your master needs to be up to date.

$ git pull

Create the branch on your local machine and switch in this branch :

$ git checkout -b [name_of_your_new_branch] Push the

branch on GitHub :

$ git push origin [name_of_your_new_branch]

When you want to commit something in your branch, be sure to be in your branch. Add -u parameter to set-upstream.

You can see all the branches created by using :

```
$ git branch -a
```

Which will show : *

approval_messages

master   master_clean

Add a new remote for your branch :

```
$ git remote add [name_of_your_remote]
[name_of_your_new_branch]
```

Push changes from your commit into your branch : $ git

push [name_of_your_new_remote] [url]

Update your branch when the original branch from official repository has been updated :

```
$ git fetch [name_of_your_remote]
```

Then you need to apply to merge changes if your branch is derivated from develop you need to do :

$ git merge [name_of_your_remote]/develop Delete a

branch on your local filesystem :

```
$ git branch -d [name_of_your_new_branch]
```

To force the deletion of local branch on your filesystem :

$ git branch -D [name_of_your_new_branch] Delete the

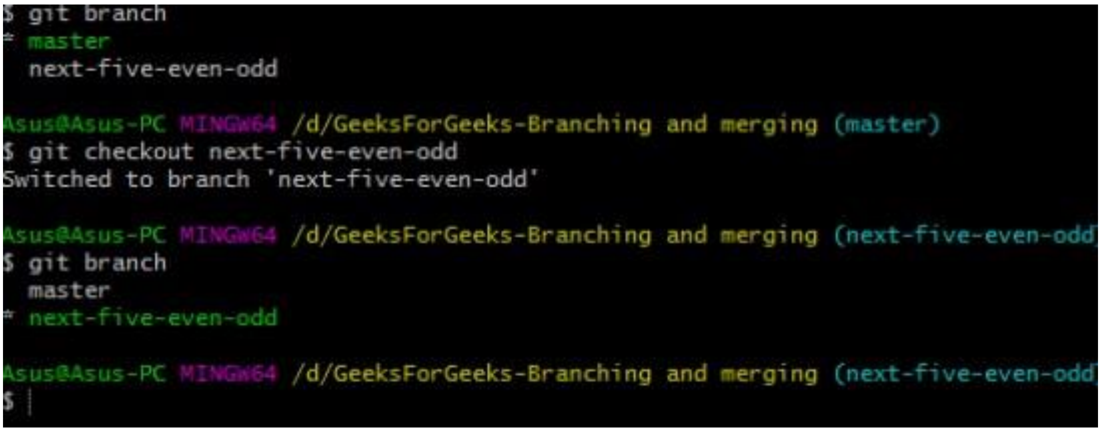
branch on GitHub :

$ git push origin :[name_of_your_new_branch]

If you want to change default branch, it's so easy with GitHub, in your fork go into Admin and in the drop-down list default branch choose what you want.

If you want create a new branch:

$ git branch <name_of_your_new_branch>

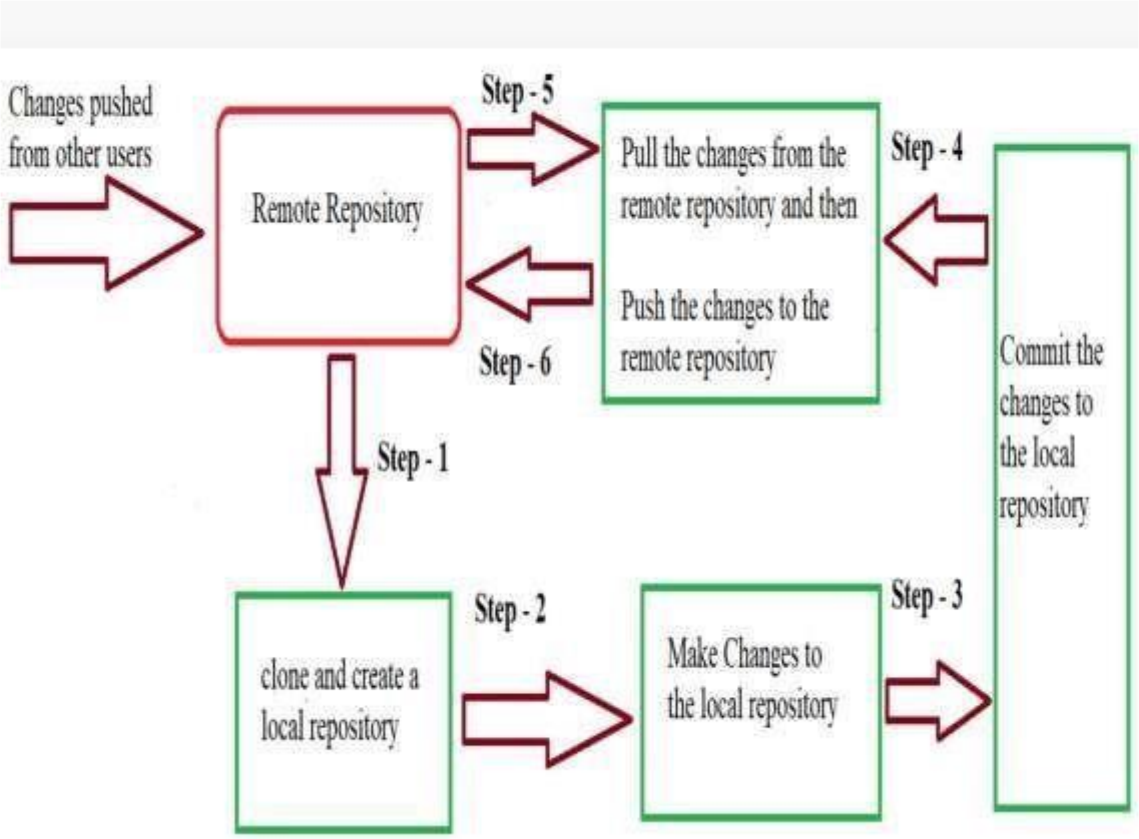


```
$ git branch
* master
  next-five-even-odd

Asus@Asus-PC MINGW64 /d/GeeksForGeeks-Branching and merging (master)
$ git checkout next-five-even-odd
Switched to branch 'next-five-even-odd'

Asus@Asus-PC MINGW64 /d/GeeksForGeeks-Branching and merging (next-five-even-odd
$ git branch
  master
* next-five-even-odd

Asus@Asus-PC MINGW64 /d/GeeksForGeeks-Branching and merging (next-five-even-odd
$ |
```

Task-5:

Aim:- Git Lifecycle description.

Git is used in our day-to-day work, we use git for keeping a track of our files, working in a collaboration with our team, to go back to our previous code versions if we face some error. Git helps us in many ways. Let us look at the Life Cycle that git has and understand more about its life cycle. Let us see some of the basic steps that we follow while working with Git –


Changes pushed from other users
Remote Repository
Step - 5
Pull the changes from the remote repository and then
Push the changes to the remote repository
Step - 6
Step - 4
Commit the changes to the local repository
Step - 1
clone and create a local repository
Step - 2
Make Changes to the local repository
Step - 3

- **In Step – 1, We first clone any of the code residing in the remote reposit ory to make our won local repository.**
- **In Step-2 we edit the files that we have cloned in our local repository an d make the necessary changes in it.**

- In Step-3 we commit our changes by first adding them to our staging ar ea and committing them with a commit message.
- In Step – 4 and Step-5 we first check whether there are any of the chang es done in the remote repository by some other users and we first pull that c hanges.
- If there are no changes we directly proceed with Step – 6 in which we p ush our changes to the remote repository and we are done with our work. When a directory is made a git repository, there are mainly 3 states which make the essence of Git Version Control System. The three states are –

- Working Directory
- Staging Area
- Git Directory

Let us understand in detail about each state.

1. Working Directory

Whenever we want to initialize our local project directory to make it a git repository, we use the git init command. After this command, git becomes aware of the files in the project although it doesn't track the files yet. The files are further tracked in the staging area.

git init

2. Staging Area

Now, to track the different versions of our files we use the command git add. We can term a staging area as a place where different versions of our files are stored. git add command copies the version of your file from your working directory to the staging area. We can, however, choose which files we need to add to the staging area because in our working directory there

are some files that we don't want to get tracked, examples include node modules, env files, temporary files, etc. Indexing in Git is the one that helps Git in understanding which files need to be added or sent. You can find your staging area in the .git folder inside the index file. // to specify which file to add to the staging area git add <filename>

// to add all files of the working directory to the staging

area git add .

3. Git Directory

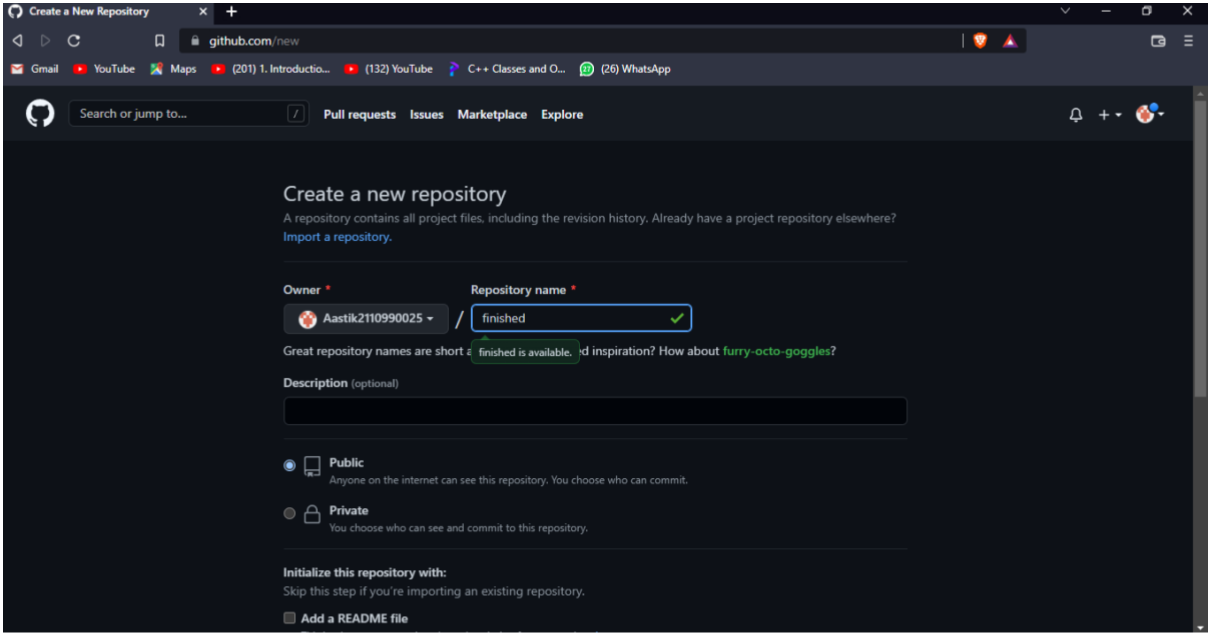
Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the git commit command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about. Git preserves the information or the metadata of the files that were committed in a Git Directory which helps Git in tracking files and basically it preserves the photocopy of the committed files. Commit also stores the name of the author who did the commit, files that are committed, and the date at which they are committed along with the commit message. git commit -m <Commit Message>

Task:6

Aim: Add collaborators on GitHub Repo

Theory:

1. Create a New Repository. A repository is like a folder for your project. You can have any number of public and private repositories in your user account. Repositories can contain folders and files, images, videos, spreadsheets, data sets, and the revision history for all files in the repository. For more information, see "About repositories." When you create a new repository, you should initialize the repository with a README file to let people know about your project. For more information, see "Creating a new repository."



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * Aastik2110990025 / Repository name * finished ✓

Great repository names are short a finished is available. d inspiration? How about furry-octo-goggles?

Description (optional)

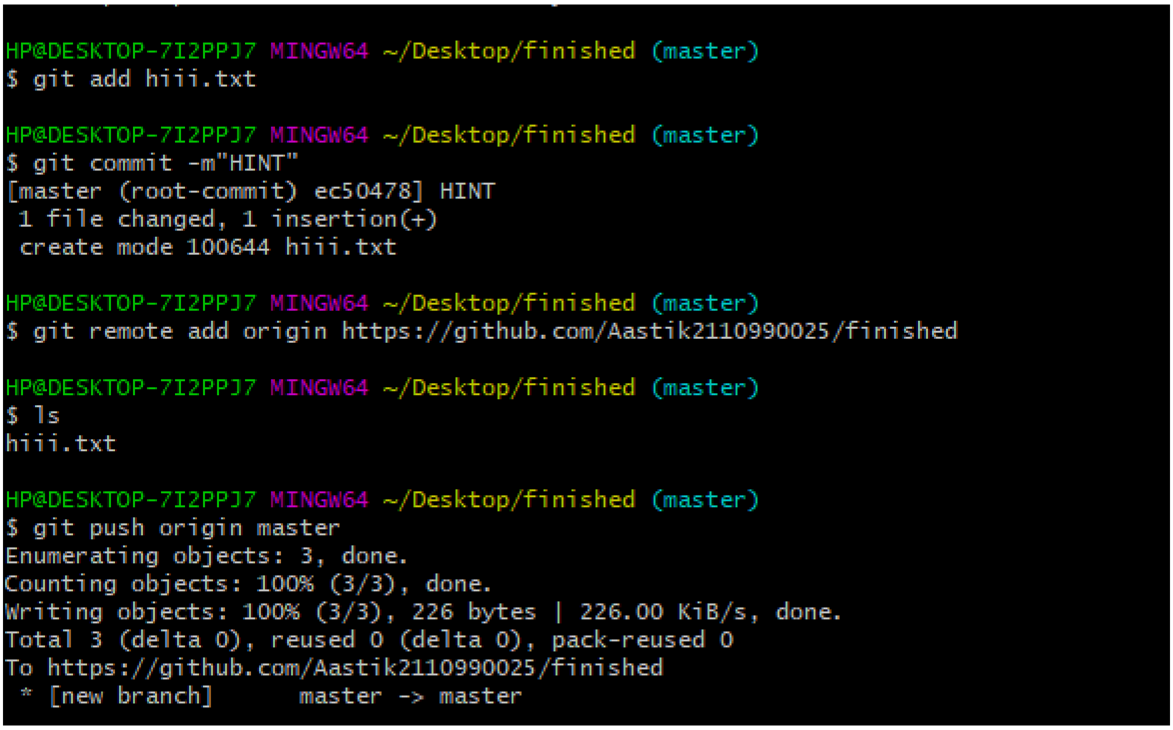
● Public
Anyone on the internet can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

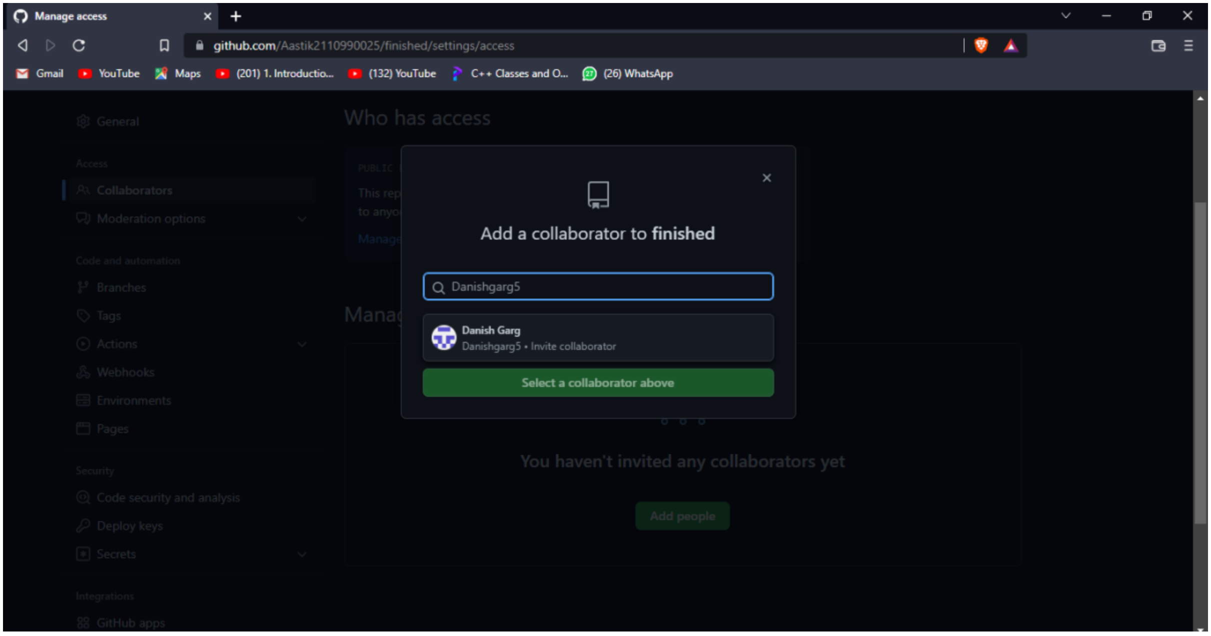
□ Add a README file

2. Now Copy the HTTP link of your repo and paste it it on your ‘Git CLI’, and merge the local repo in remote repo.



```
HP@DESKTOP-7I2PPJ7 MINGW64 ~/Desktop/finished (master)
$ git add hiii.txt

HP@DESKTOP-7I2PPJ7 MINGW64 ~/Desktop/finished (master)
$ git commit -m"HINT"
[master (root-commit) ec50478] HINT
 1 file changed, 1 insertion(+)
 create mode 100644 hiii.txt

HP@DESKTOP-7I2PPJ7 MINGW64 ~/Desktop/finished (master)
$ git remote add origin https://github.com/Aastik2110990025/finished

HP@DESKTOP-7I2PPJ7 MINGW64 ~/Desktop/finished (master)
$ ls
hiii.txt

HP@DESKTOP-7I2PPJ7 MINGW64 ~/Desktop/finished (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 226 bytes | 226.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Aastik2110990025/finished
 * [new branch]      master -> master
```

3. Go to Collaborators in Repo Setting, Add the username or email of the Collaborator you want to add to your Repo.



github.com/Aastik2110990025/finished/settings/access

Add a collaborator to **finished**

Danishgarg5

Danish Garg
Danishgarg5 • Invite collaborator

Select a collaborator above

You haven't invited any collaborators yet

4. Invitation Mail is sent to the Collaborator, the collaborator has to accept this Invitation.

5. **New collaborator has now access to the Aastik2110990025 repo.**

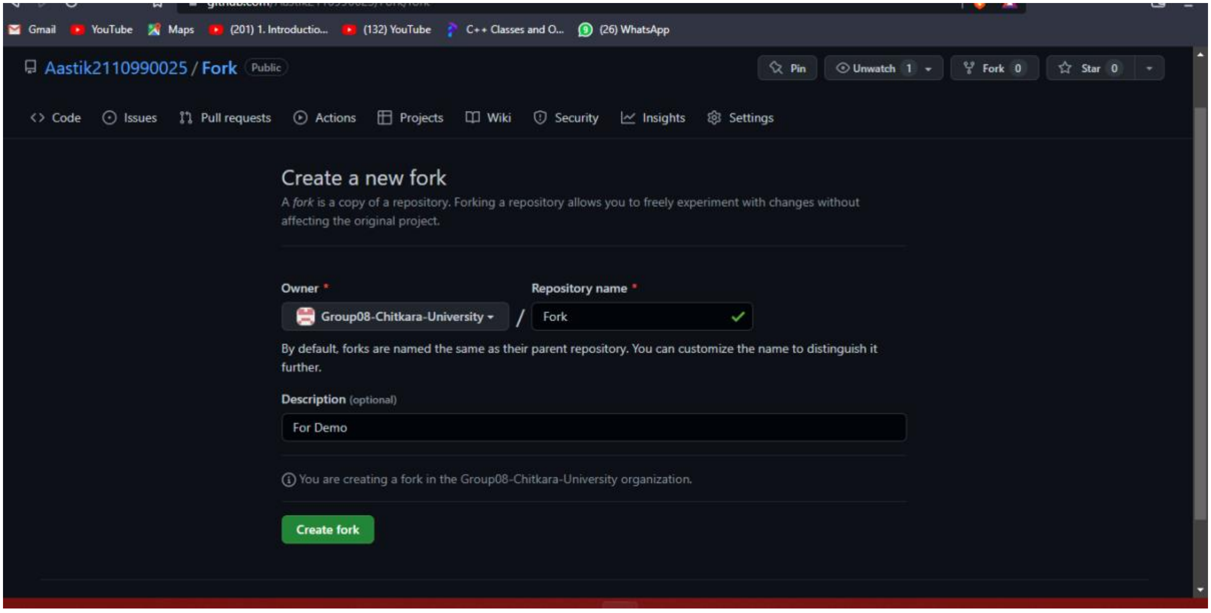
Task:7

Aim: Fork and Commit

Theory:

Fork:- A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original Project.

1. To fork a repository first thing, you need to do is, sign in to your GitHub account and then you came to the repository you want to fork, so here for demo purposes am using the repository.



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Owner * Group08-Chitkara-University / Repository name * Fork ✓

By default, forks are named the same as their parent repository. You can customize the name to distinguish it further.

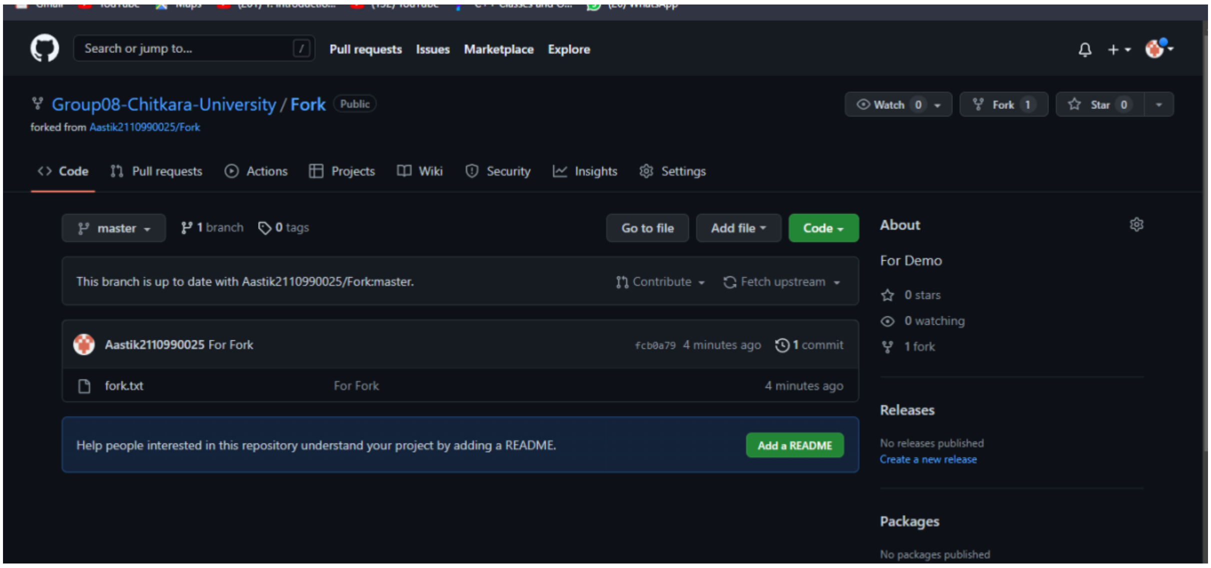
Description (optional)

For Demo

ⓘ You are creating a fork in the Group08-Chitkara-University organization.

Create fork

2. Click on the Fork button on the right upside corner. Then it will ask to create a new fork, add a description if you want and then click on create a fork.



forked from Aastik2110990025/Fork

This branch is up to date with Aastik2110990025/Fork:master.

Aastik2110990025 For Fork fcb0a79 4 minutes ago 1 commit

fork.txt For Fork 4 minutes ago

Help people interested in this repository understand your project by adding a README.

For Demo

No releases published
Create a new release

No packages published

3. Now you will have a copy of the repo you have forked from other user. Now you can do any modification you want without making changes to main source code.
4. Now type https://github.com/Group08-Chitkara-University/Fork git on CLI.
   Git pull --> This command is used to fetch the remote repo or to clone the repo.
5. Now Open the file make changes in it and commit it and push it to remote.
6. Now go to GitHub and accept the merge request.

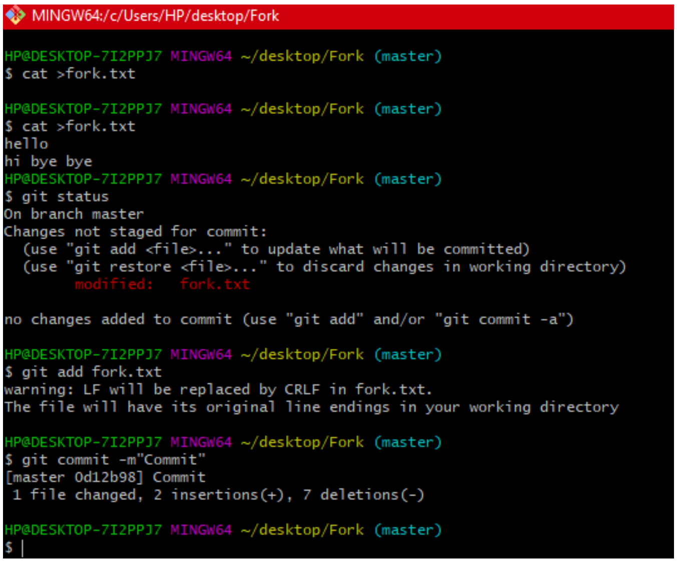
Task:8

Aim:- Merge and Resolve Conflicts.

Theory:

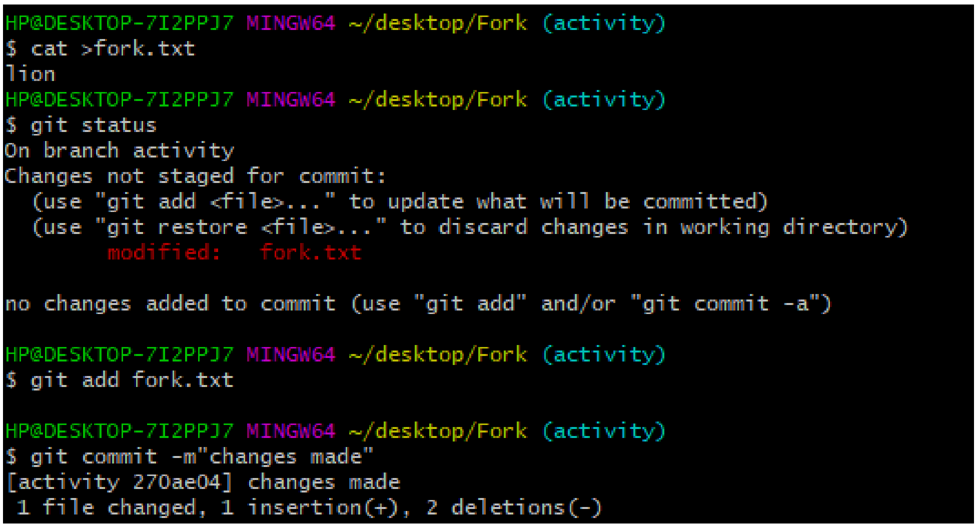
1. **Do changes in the master branch and commit those changes. And check out to a different branch and again do changes and commit it. Now checkout to the master branch and merge that branch into master.**

COMMIT IN MASTER BRANCH



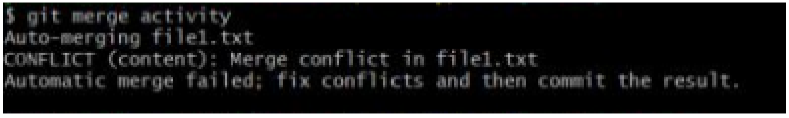
```
HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (master)
$ cat >fork.txt

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (master)
$ cat >fork.txt
hello
hi bye bye
HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   fork.txt

no changes added to commit (use "git add" and/or "git commit -a")

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (master)
$ git add fork.txt
warning: LF will be replaced by CRLF in fork.txt.
The file will have its original line endings in your working directory

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (master)
$ git commit -m"Commit"
[master 0d12b98] Commit
 1 file changed, 2 insertions(+), 7 deletions(-)

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (master)
$ |
```

COMMIT IN ACTIVITY BRANCH



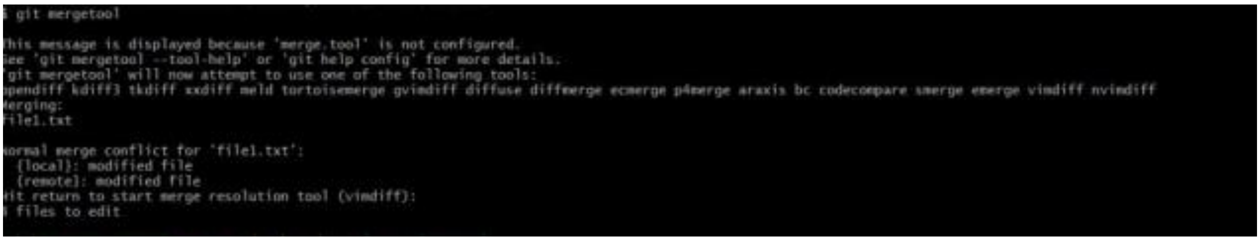
```
HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (activity)
$ cat >fork.txt
lion
HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (activity)
$ git status
On branch activity
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   fork.txt

no changes added to commit (use "git add" and/or "git commit -a")

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (activity)
$ git add fork.txt

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Fork (activity)
$ git commit -m"changes made"
[activity 270ae04] changes made
 1 file changed, 1 insertion(+), 2 deletions(-)
```

2. Now try to merge it will give conflict error.



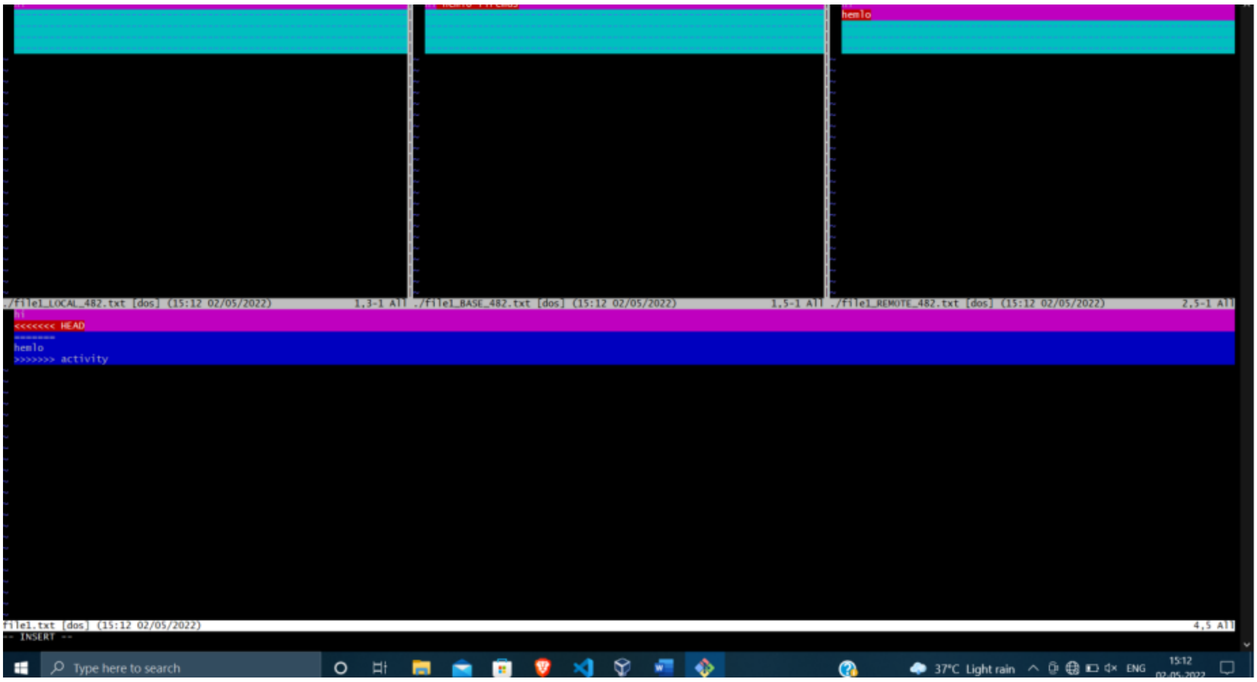
```
$ git merge activity
Auto-merging file1.txt
CONFLICT (content): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

3. Use Command “git mergetool” to solve the conflict. git -mergetool – Run merge conflict resolution tools to resolve merge conflicts.



```
$ 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:
file1.txt

Normal merge conflict for 'file1.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
4 files to edit
```

4. Press “I” to insert, after insertion. Press “:wq”. The merge conflict is solved and our activity branch is merged to master branch.



```
hemlo
~
./file1_LOCAL_482.txt [dos] (15:12 02/05/2022)       1,3-1 All ./file1_BASE_482.txt [dos] (15:12 02/05/2022)       1,5-1 All ./file1_REMOTE_482.txt [dos] (15:12 02/05/2022)       2,5-1 All
hi
<<<<<<< HEAD
=======
hemlo
>>>>>>> activity
~
file1.txt [dos] (15:12 02/05/2022)                                                                                    4,5 All
-- INSERT --
```

Task: -9

Aim: Reset and Revert

Theory:

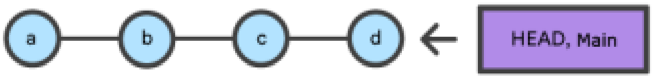
Git-revert – Revert some existing commits.

1. On Git Bash CLI, Type command “git revert”. It reverts the changes that done before Commit.

git revert HEAD~3:- Revert the changes specified by the fourth last commit in HEAD and create a new commit with the reverted changes

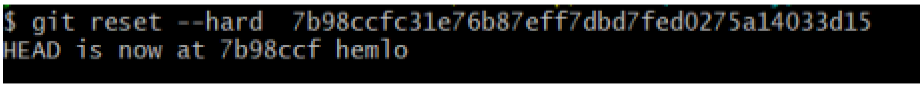
Git-reset - Reset current HEAD to the specified state

2. At a surface level, git reset is similar in behaviour to git checkout. Where git checkout solely operates on the HEAD ref pointer, git reset will move the HEAD ref pointer and the current branch ref pointer. To better demonstrate this behaviour, consider the following example. This example demonstrates a sequence of commits on the main branch. The HEAD ref and main branch ref currently point to commit d. Now let us execute and compare, both git checkout b and git reset b.



GIT RESET:

1. reset is the command we use when we want to move the repository back to a previous commit, discarding any changes made after that commit.

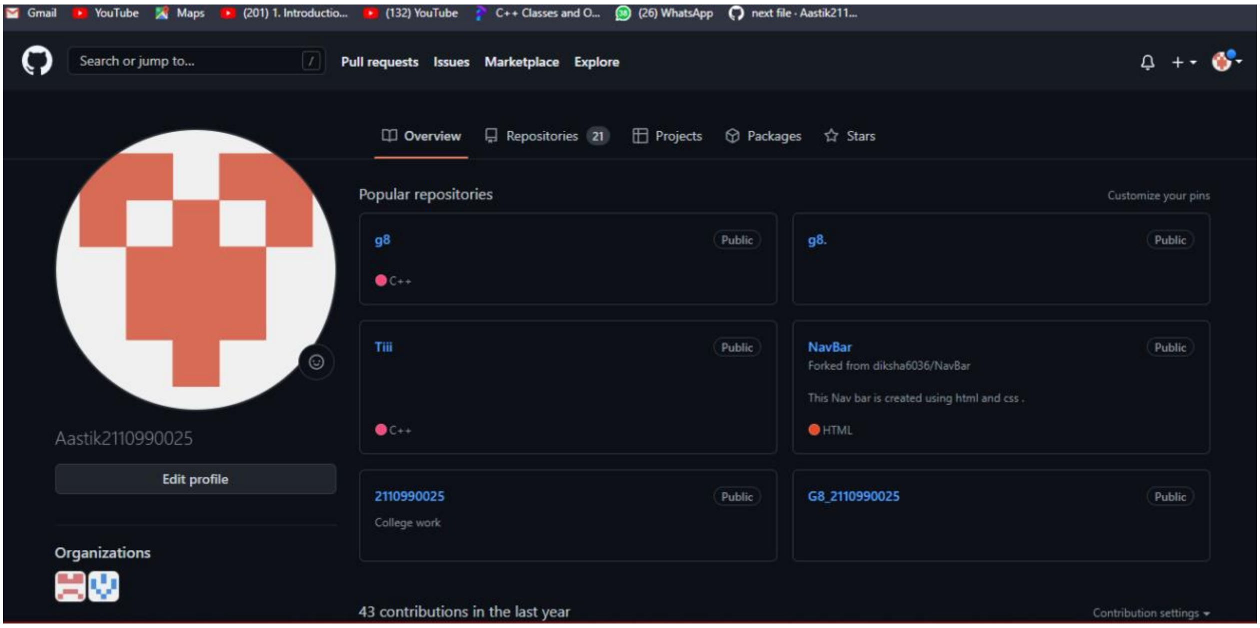


```
$ git reset --hard  7b98ccfc31e76b87eff7dbd7fed0275a14033d15
HEAD is now at 7b98ccf hemlo
```

PROJECT WORK

TASK-2

1. Login to your Github account and you will land on the homepage as shown below. Click on Repositories option in the menu bar.



Aastik2110990025

Edit profile

Organizations

Popular repositories

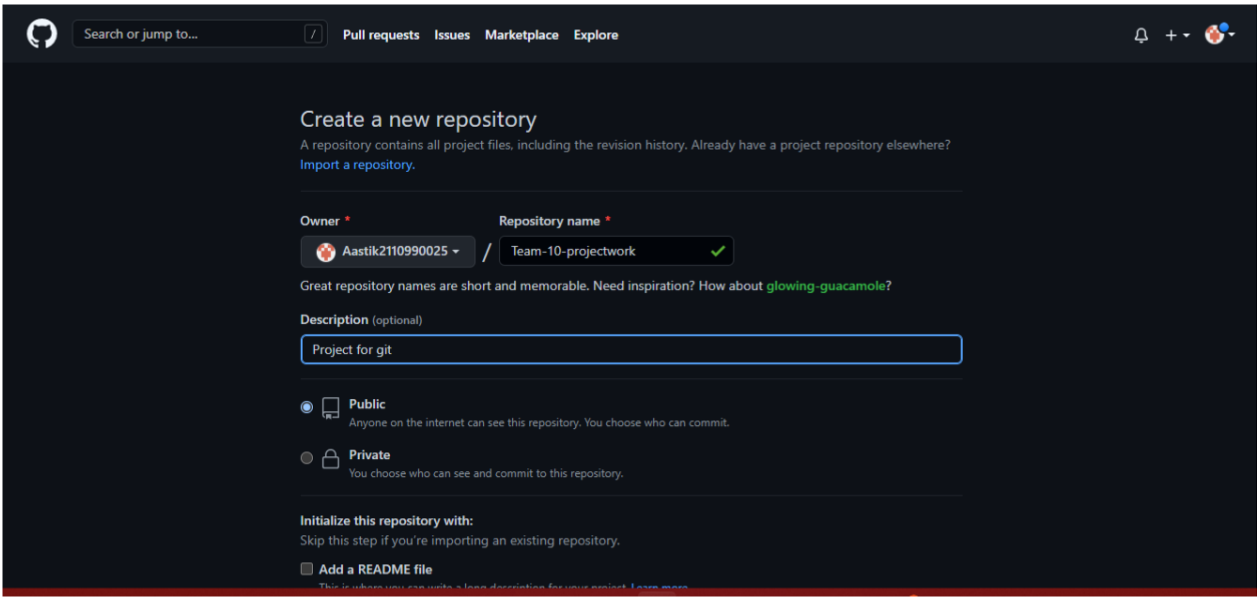
Customize your pins

g8 Public C++	g8. Public
Tiii Public C++	NavBar Public Forked from diksha6036/NavBar This Nav bar is created using html and css . HTML
2110990025 Public College work	G8_2110990025 Public

43 contributions in the last year

Contribution settings

2. Enter the Repository name and add the description of the repository. Select if you want the repository to be public or private.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * Aastik2110990025 / Repository name * Team-10-projectwork

Great repository names are short and memorable. Need inspiration? How about glowing-guacamole?

Description (optional)

Project for git

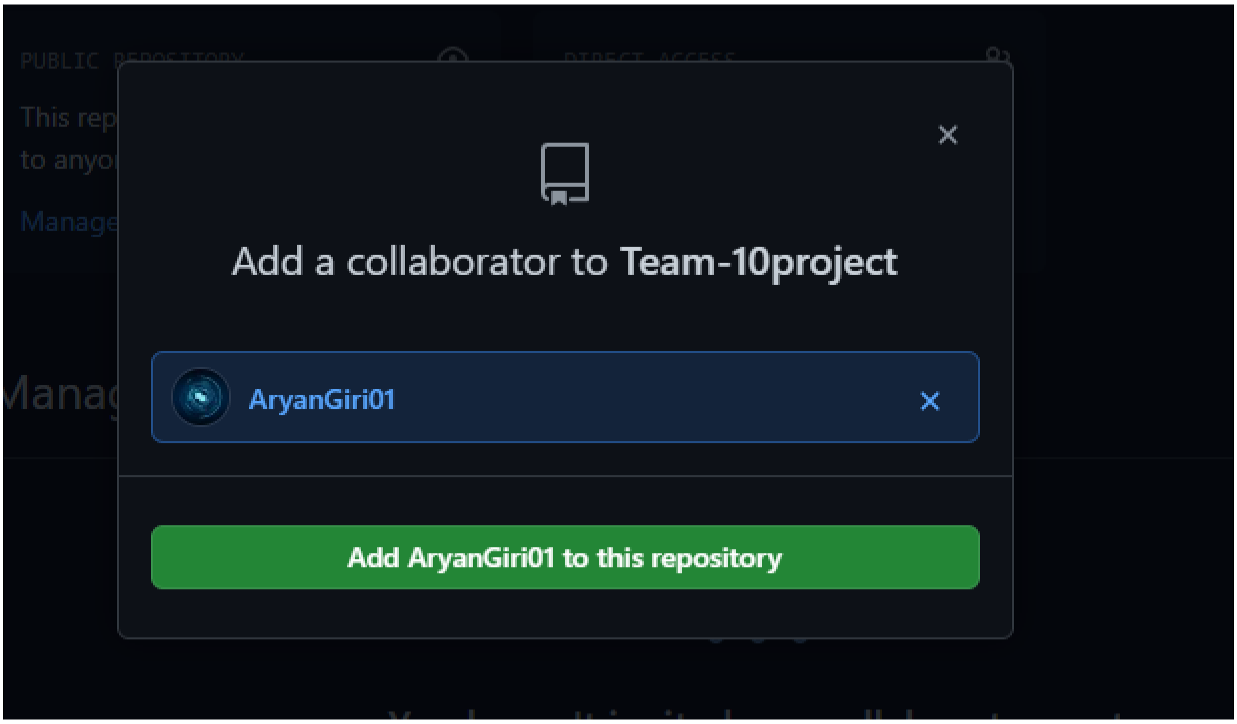
● Public
Anyone on the internet can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

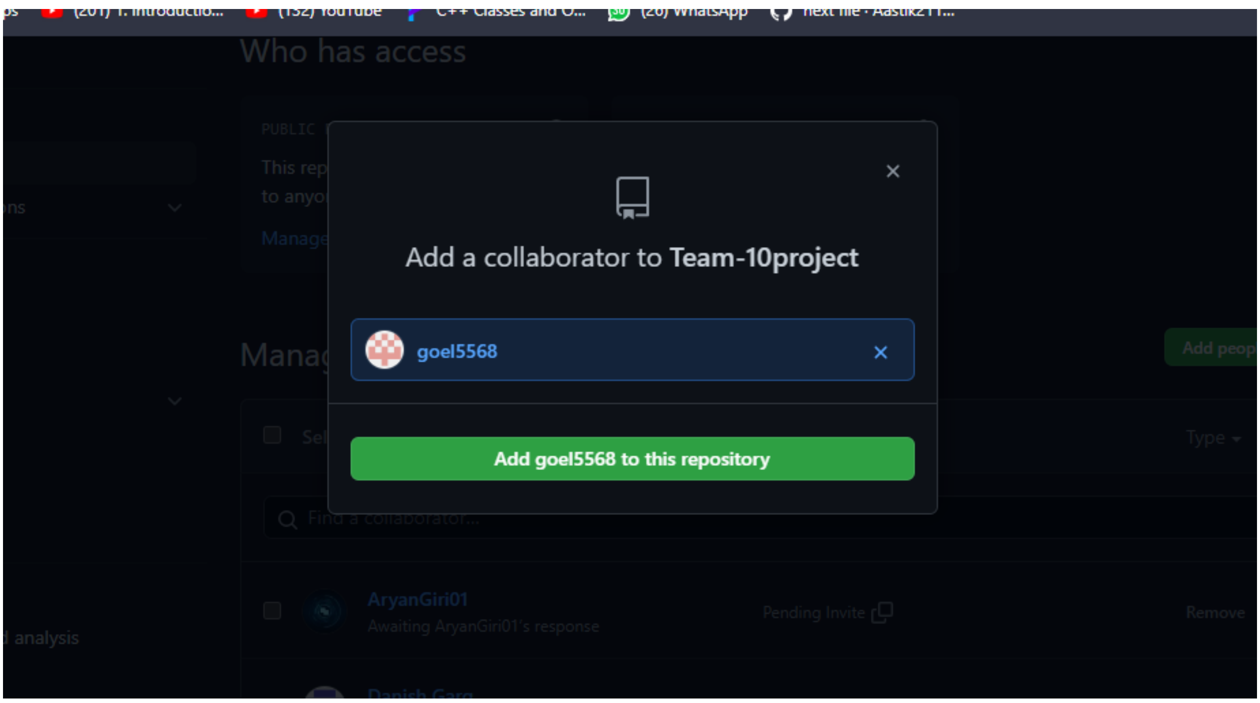
□ Add a README file

3. Now add collaborators in the repository.



Add a collaborator to Team-10project

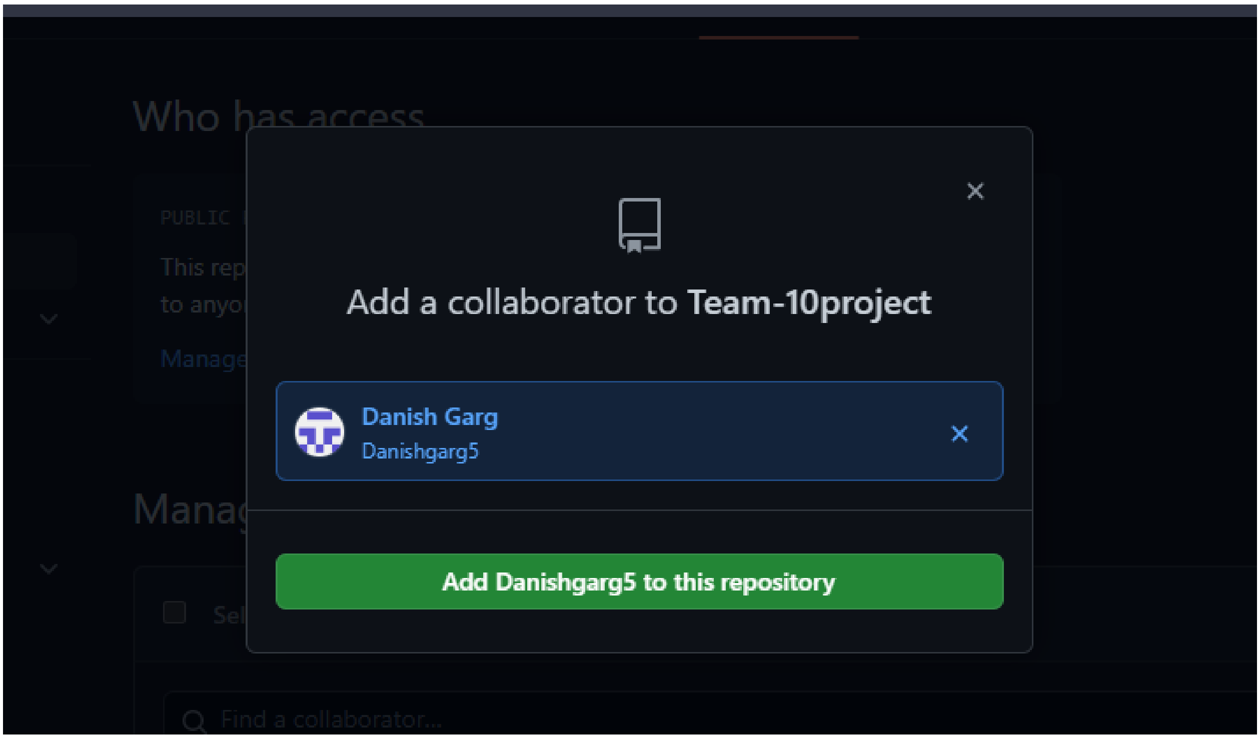
AryanGiri01



Add a collaborator to Team-10project

goel5568

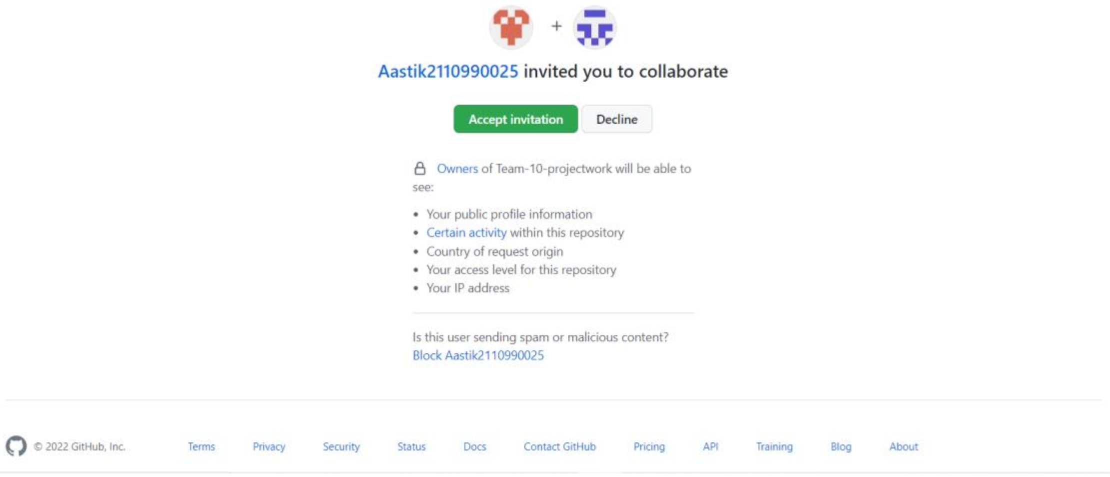
AryanGiri01
Awaiting AryanGiri01's response
Pending Invite



Add a collaborator to **Team-10project**

Danish Garg
Danishgarg5

4. Now the collaborators will accept the invite that was sent to them.



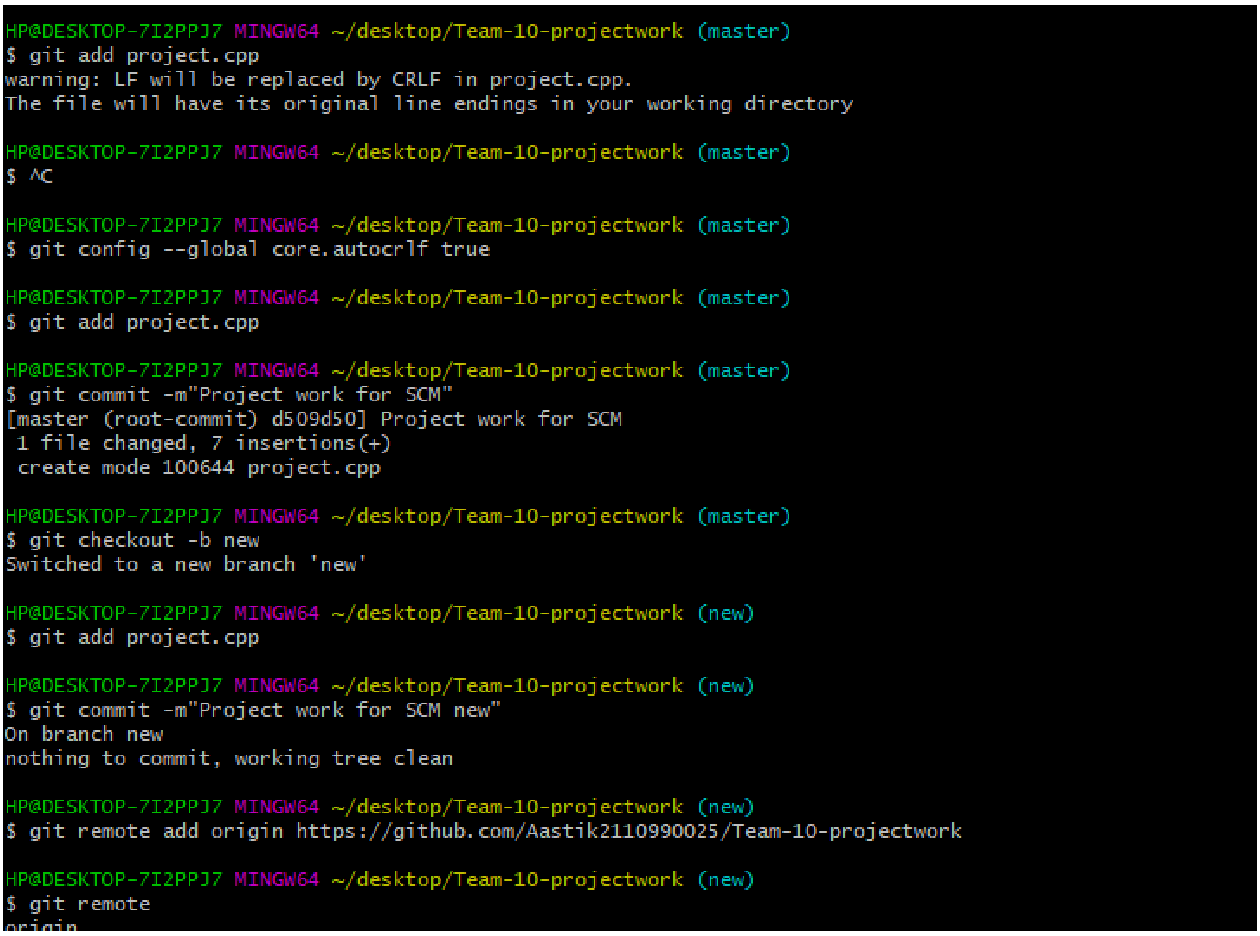
Aastik2110990025 invited you to collaborate

Owners of Team-10-projectwork will be able to see:

- Your public profile information
- Certain activity within this repository
- Country of request origin
- Your access level for this repository
- Your IP address

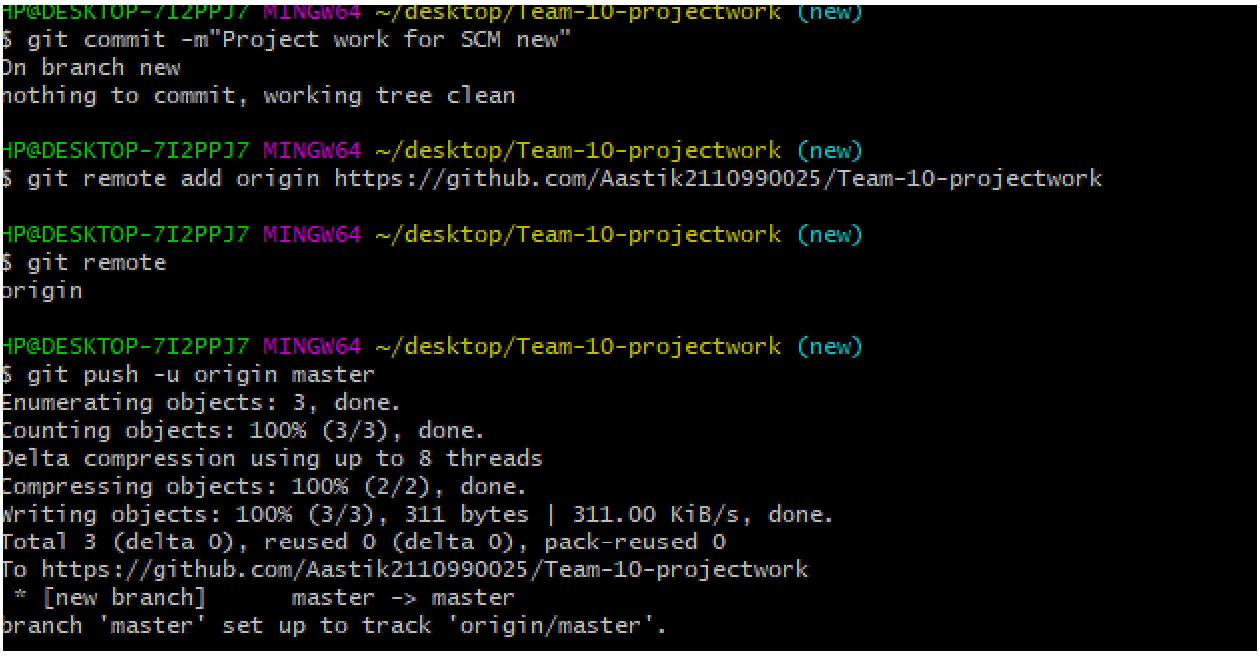
Is this user sending spam or malicious content?
Block Aastik2110990025

5. To open a pull request we first have to make a new branch, by using **git branch branchname** option. After making new branch we add a file to the branch or make changes in the existing file. Add and commit the changes to the local repository.



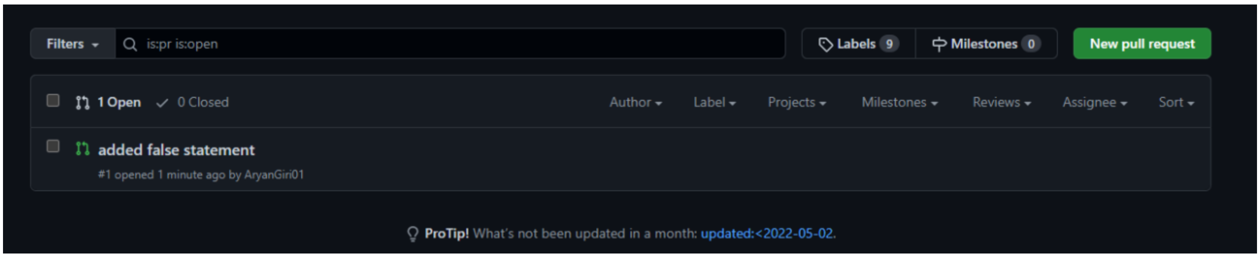
```
HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (master)
$ git add project.cpp
warning: LF will be replaced by CRLF in project.cpp.
The file will have its original line endings in your working directory

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (master)
$ ^C

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (master)
$ git config --global core.autocrlf true

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (master)
$ git add project.cpp

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (master)
$ git commit -m"Project work for SCM"
[master (root-commit) d509d50] Project work for SCM
 1 file changed, 7 insertions(+)
 create mode 100644 project.cpp

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (master)
$ git checkout -b new
Switched to a new branch 'new'

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (new)
$ git add project.cpp

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (new)
$ git commit -m"Project work for SCM new"
On branch new
nothing to commit, working tree clean

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (new)
$ git remote add origin https://github.com/Aastik2110990025/Team-10-projectwork

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (new)
$ git remote
origin
```

6. Use **git push origin branch name** option to push the new branch to the main repository.



```
HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (new)
$ git commit -m"Project work for SCM new"
On branch new
nothing to commit, working tree clean

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (new)
$ git remote add origin https://github.com/Aastik2110990025/Team-10-projectwork

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (new)
$ git remote
origin

HP@DESKTOP-7I2PPJ7 MINGW64 ~/desktop/Team-10-projectwork (new)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 311 bytes | 311.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Aastik2110990025/Team-10-projectwork
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

7. Now collaborator makes changes and creates a pull request.
8. Now the After opening a pull request all the team members will be sent the request if they want to merge or close the request. If the team member chooses not to merge your pull request, they will close you're the pull request. To close the pull request simply click on close pull request and add comment/ reason why you closed the pull request.

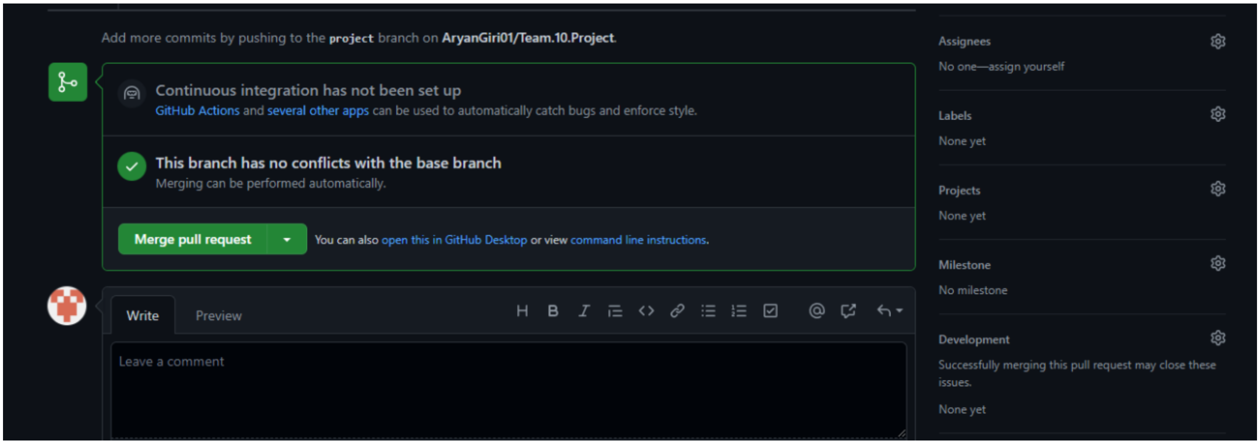


added false statement

#1 opened 1 minute ago by AryanGiri01

ProTip! What's not been updated in a month: updated:<2022-05-02.

9. Now Do the required changes in the repository, add and commit these changes in the local repository in a new branch. Push the modified branch using **git push -u origin branchname**. Open a pull request by following the procedure from the above experiment. The pull request will be created and will be visible to all the team members. Ask your team member to login to his/her Github account. They will notice a new notification in the pull request menu. Click on it. The pull request generated by you will be visible to them.


Add more commits by pushing to the project branch on AryanGiri01/Team.10.Project.

Continuous integration has not been set up

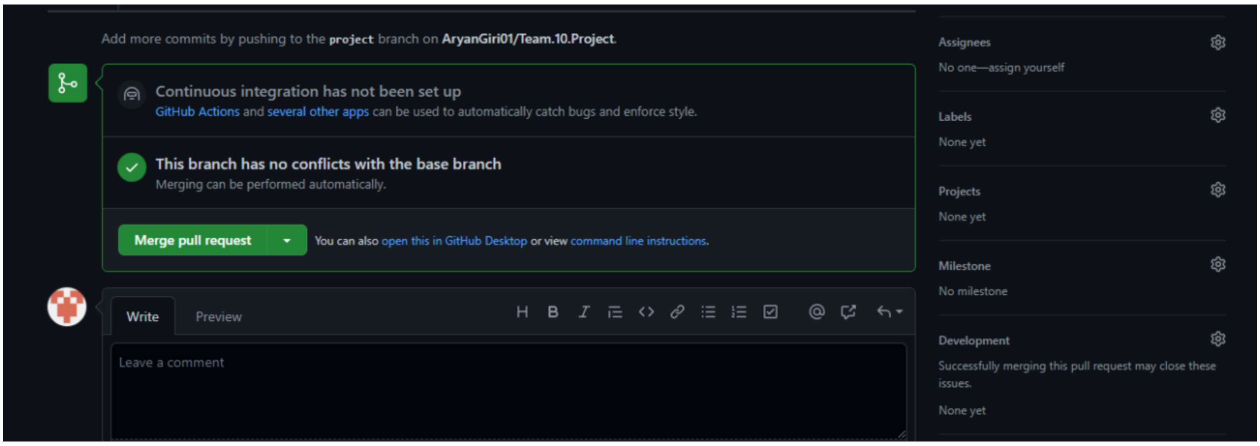
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch

Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

10. Click on the pull request. Two option will be available, either to close the pull request or Merge the request with the main branch. By selecting the merge branch option the main branch will get updated for all the team members. By selecting close the pull request the pull request is not accepted and not merged with main branch.



Add more commits by pushing to the `project` branch on AryanGiri01/Team.10.Project.

Continuous integration has not been set up

GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch

Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

Assignees: No one—assign yourself

Labels: None yet

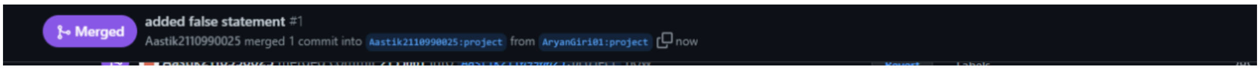
Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues.

None yet

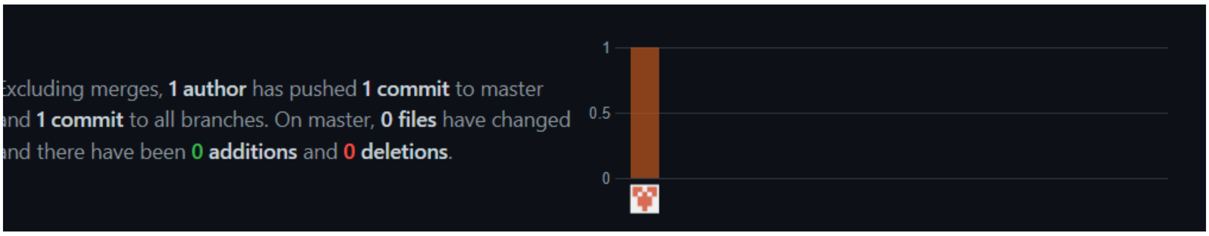
11. If the file is merged it will show a message to you.



Merged added false statement #1

Aastik2110990025 merged 1 commit into `Aastik2110990025:project` from `AryanGiri01:project` now

12. Network Graph need to be recorded.



Excluding merges, **1 author** has pushed **1 commit** to master and **1 commit** to all branches. On master, **0 files** have changed and there have been **0 additions** and **0 deletions**.