Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **CSE**

Submitted By:
Abhishek Kumar Gupta
2110990066
G01

Submitted To:
Monit Kapoor

CHITKARA UNIVERSITY

INDEX

What is GIT and why is it used?

Git is a source code management technology used by DevOps. Git is a piece of software that allows you to track changes in any group of files. It is a free and open-source version control system that may be used to efficiently manage small to big projects. Git is a version control system that allows numerous developers to collaborate on non-linear development projects.
Git is an example of a distributed version control system (DVCS) (hence Distributed Version Control System).

What is GITHUB?

GitHub is a version management and collaboration tool for programming. It allows you and others to collaborate on projects from any location.

What is the difference between GIT and GITHUB?

Git	GitHub
1. It is a software	1. It is a service
2. It is installed locally on the system	2. It is hosted on Web
3. It is a command line tool	3. It provides a graphical interface
4. It is a tool to manage different versions of edits, made to files in a git repository	4. It is a space to upload a copy of the Git repository
5. It provides functionalities like Version Control System Source Code Management	5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features

What is Repository?

A repository stores all of your project's files, as well as the revision history for each one. Within the repository, you may discuss and monitor your project's progress. The.git/ subdirectory within a project is a Git repository. This repository keeps track of any changes made to files in your project over time, creating a history. That is, if you delete the.git/ subdirectory, you are also deleting the history of your project.

What is Version Control System (VCS)?

Version Control Systems are the software tools for tracking/managing all the changes made to the source code during the project development. It keeps a record of every single change made to the code. It also allows us to turn back to the previous version of the code if any mistake is made in the current version. Without a VCS in place, it would not be possible to monitor the development of the project.

Types of VCS

- Local Version Control System

- Centralized Version Control System
- Distributed Version Control System

I. **Local Version Control System:** Local Version Control System is located in your local machine. If the local machine crashes, it would not be possible to retrieve the files, and all the information will be lost. If anything happens to a single version, all the versions made after that will be lost.

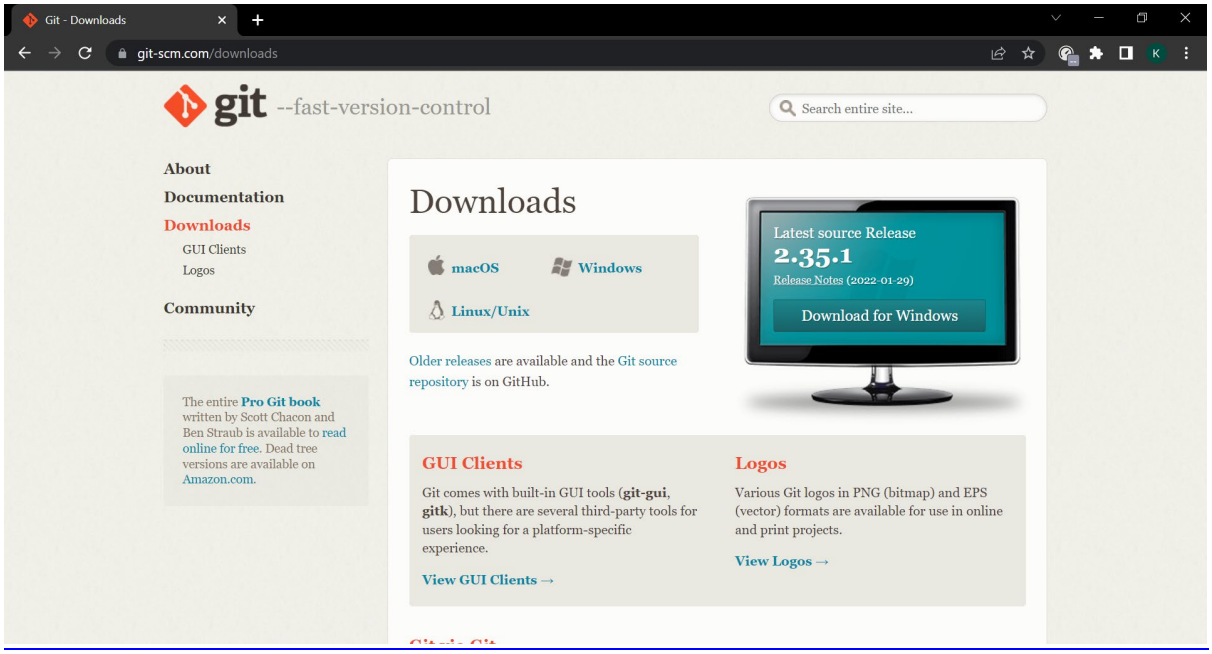
II. **Centralized Version Control System:** In the Centralized Version Control Systems, there will be a single central server that contains all the files related to the project, and many collaborators checkout files from this single server (you will only have a working copy). The problem with the Centralized Version Control Systems is if the central server crashes, almost everything related to the project will be lost.

III. **Distributed Version Control System:** In a distributed version control system, there will be one or more servers and many collaborators similar to the centralized system. But the difference is, not only do they check out the latest version, but each collaborator will have an exact copy of the main repository on their local machines. Each user has their own repository and a working copy. This is very useful because even if the server crashes we would not lose everything as several copies are residing in several other computers.

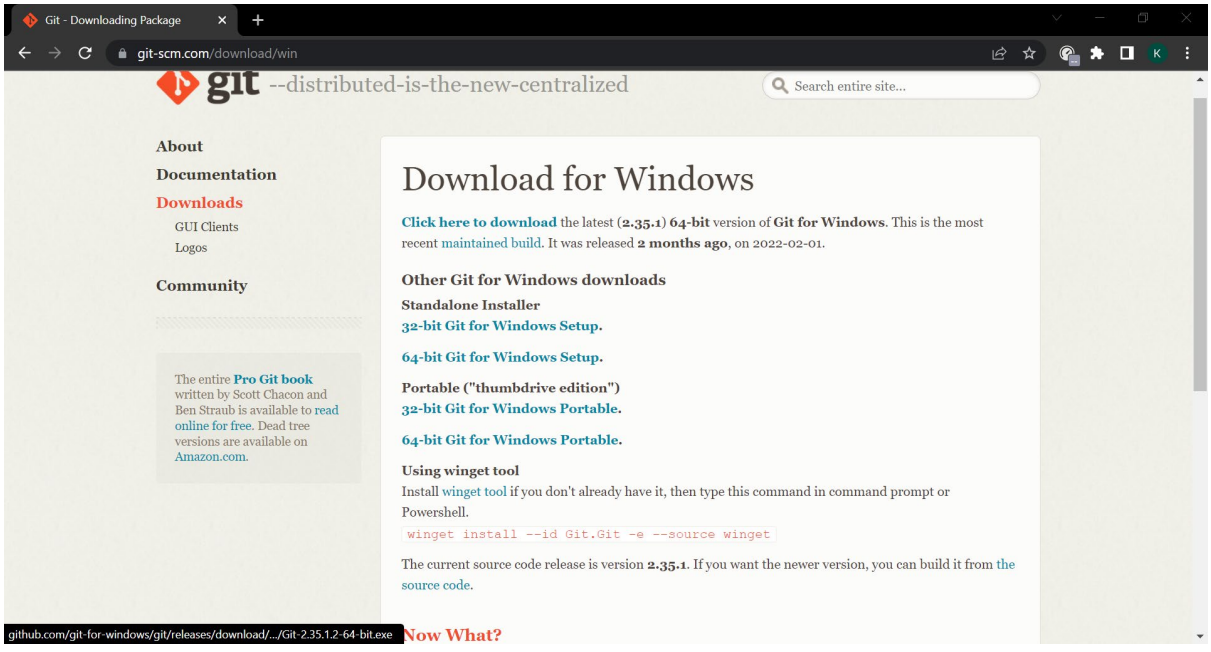
Experiment No. 01

Aim: Setting up of Git Client

✧ For git installation on your system, go to the linked URL.
https://git-scm.com/downloads

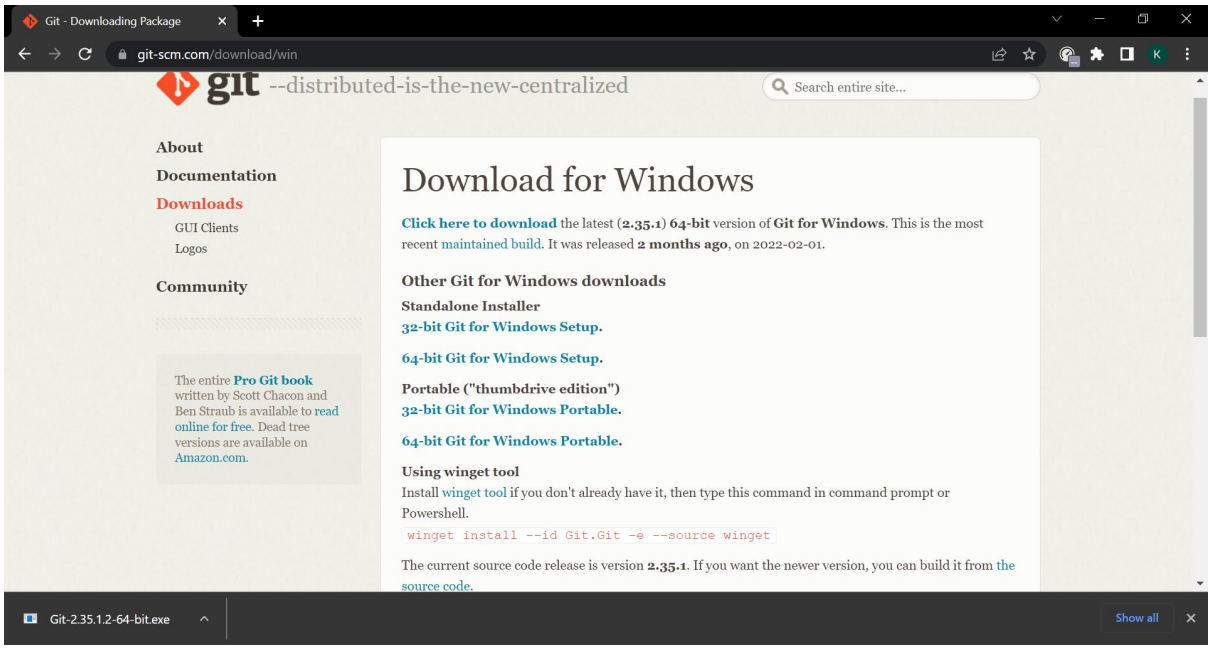


✧ You must first access this webpage and then choose your operating system by clicking on it. I'll walk you through the processes for the Windows operating system in this article.

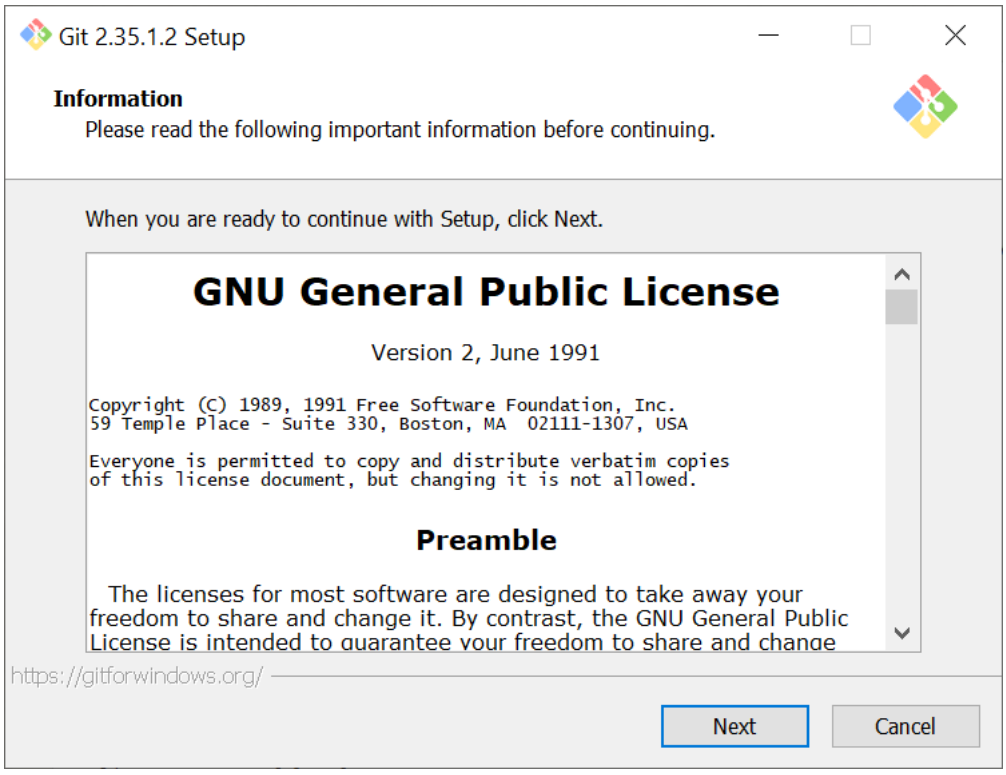


✧ Select the CPU for your system now. (Most of the system now runs on

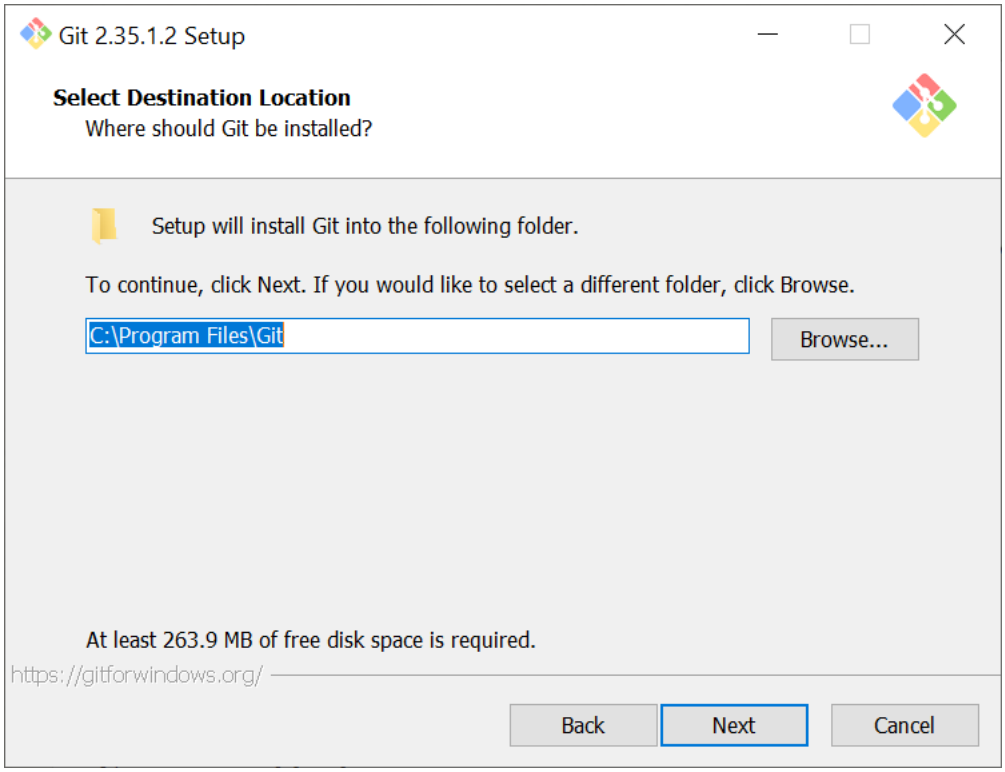
64-bit processors.) Your download will begin when you pick a processor.



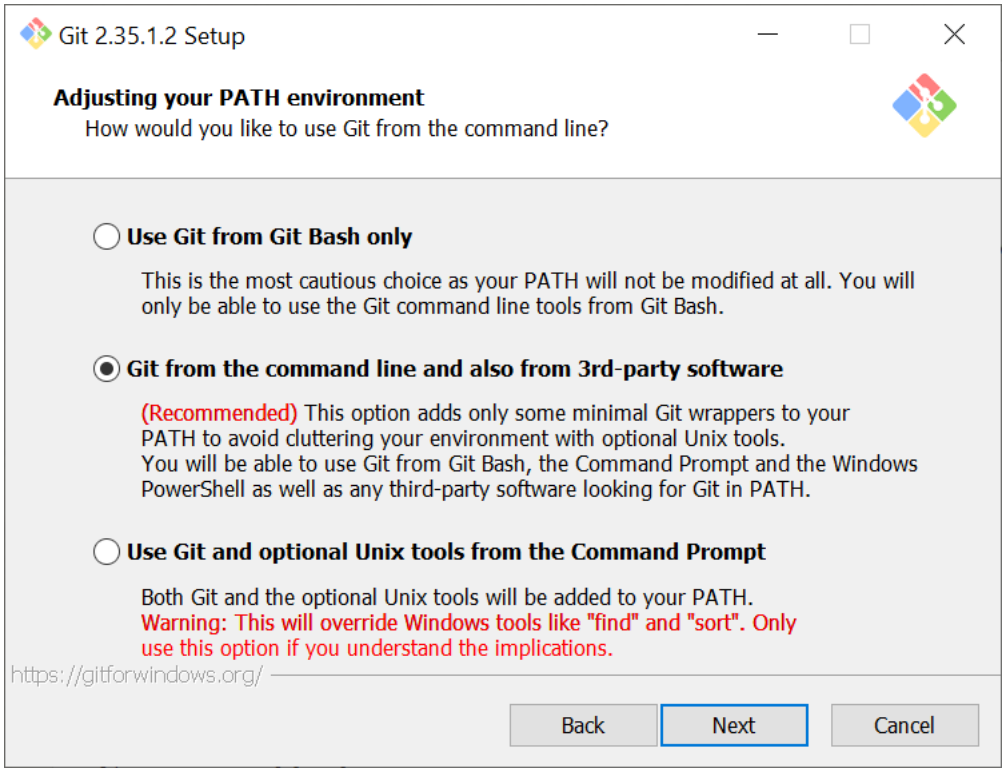
- ✧ You must now open the Git folder.
- ✧ You will be asked if you want to enable this programe to make modifications to your PC once you launch it.
- ✧ YES should be selected.



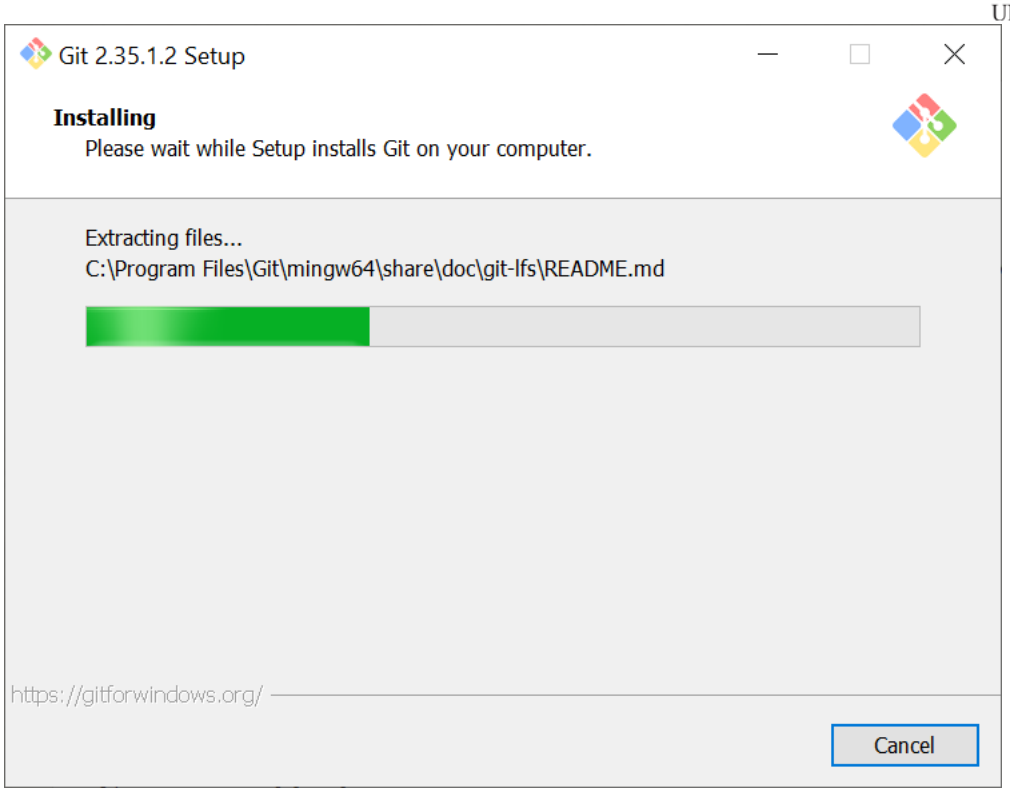
✧ Click on Next



✧ Continue clicking on next few times more

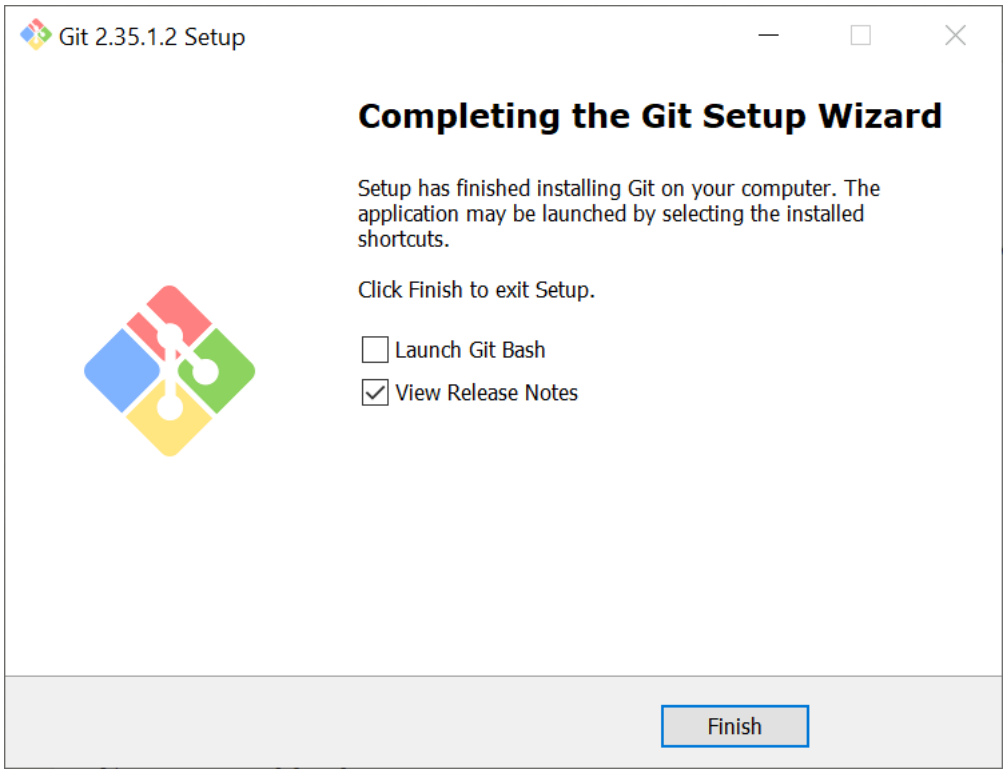


✧ Now select the Install option.



✧ Click on Finish after the installation is finished.

The installation of the git is finished and now we have to setup git client and GitHub account.

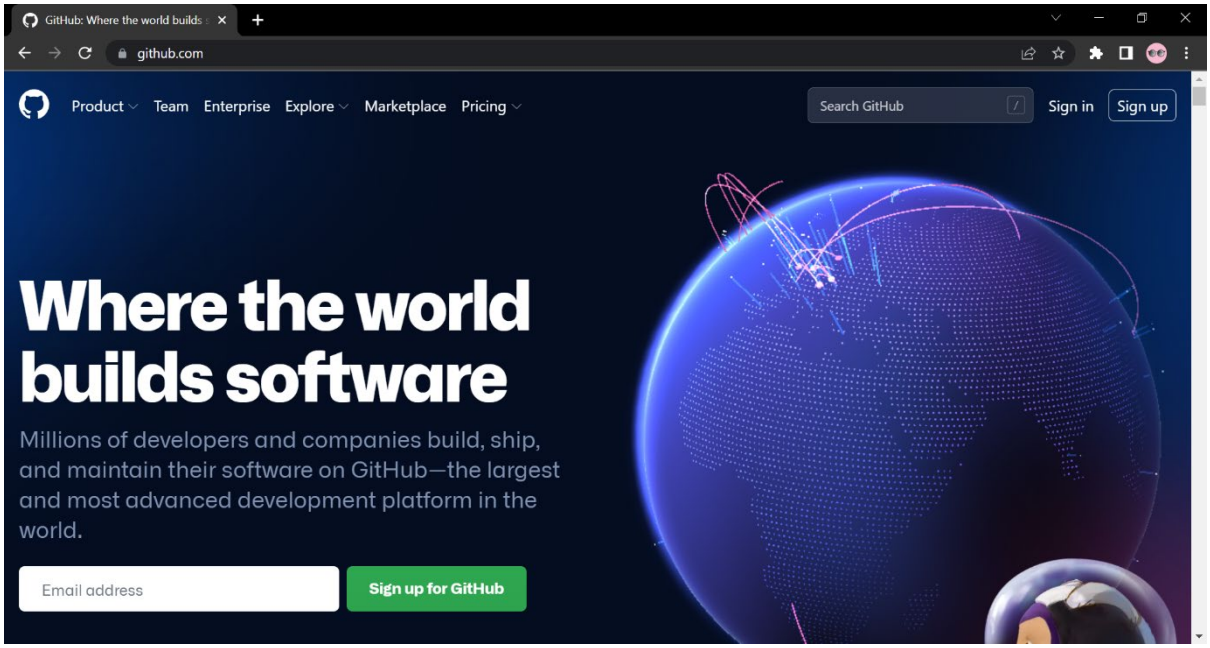


Experiment No. 02

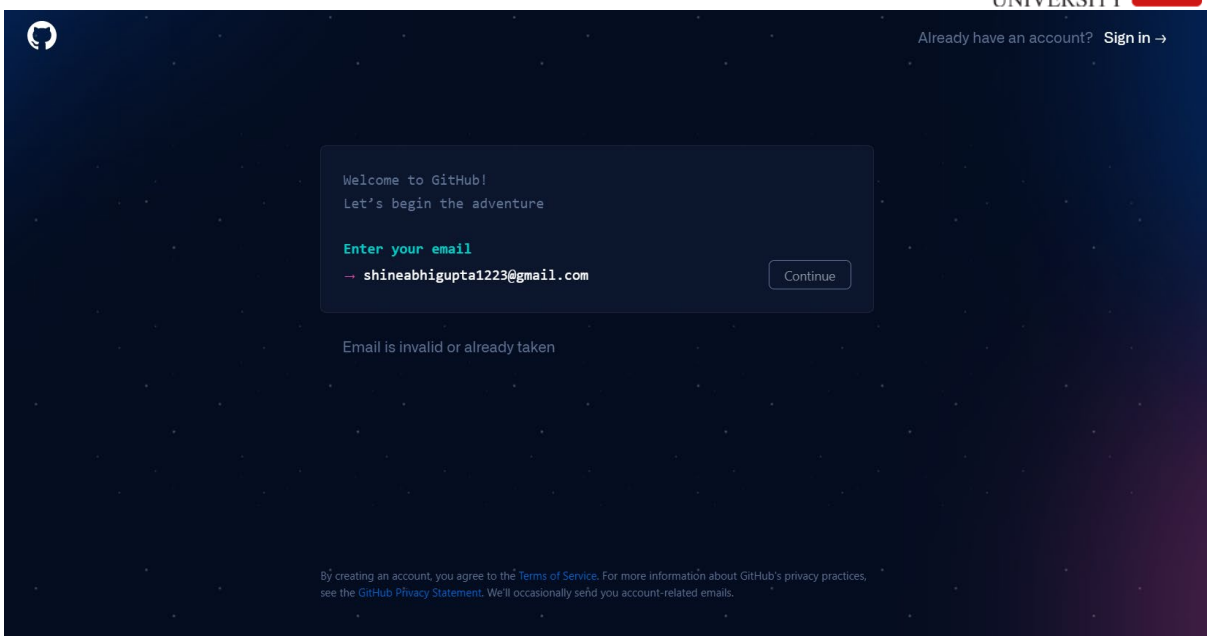
Aim: Setting up GitHub Account

✧ Open your web browser search GitHub login.

✧ Click on Create an account if you are a new user or if you have already an account, please login.



✧ After clicking on "Create a New Account," you will be sent to a new page where you must enter your email address for your account. Now type in the password you'd want to use for your GitHub account. Then you'll be prompted to enter your username.

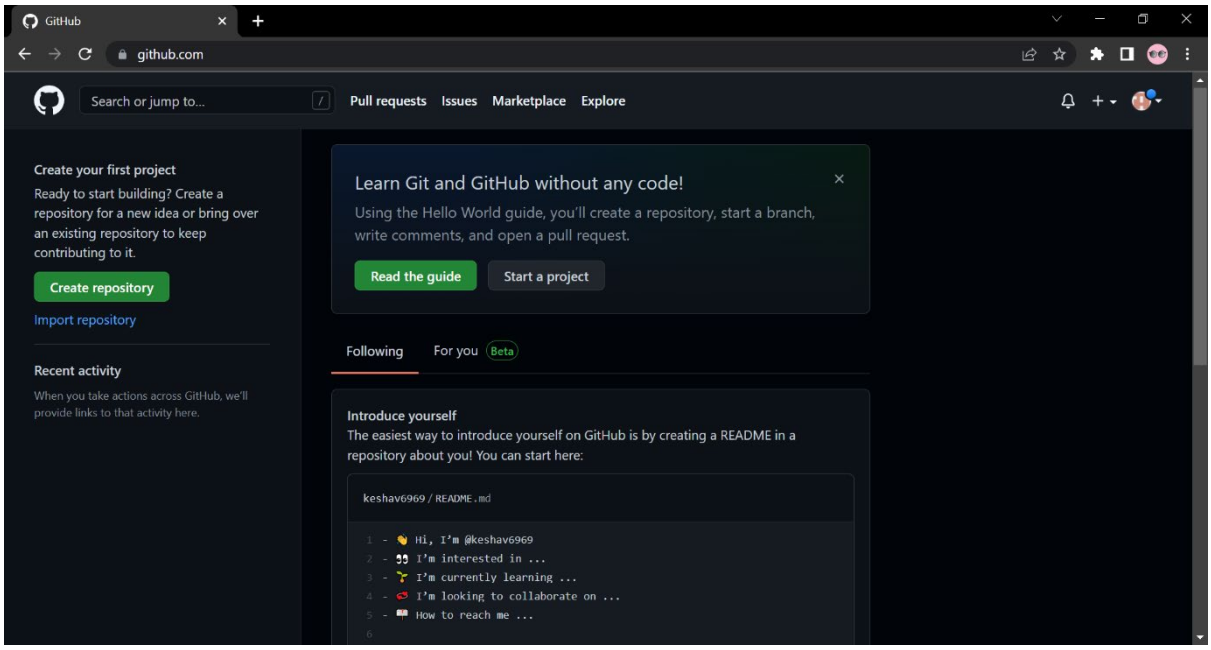


✧ Now Click on Create Account.

shineabhigupta1223@gmail.com

✧

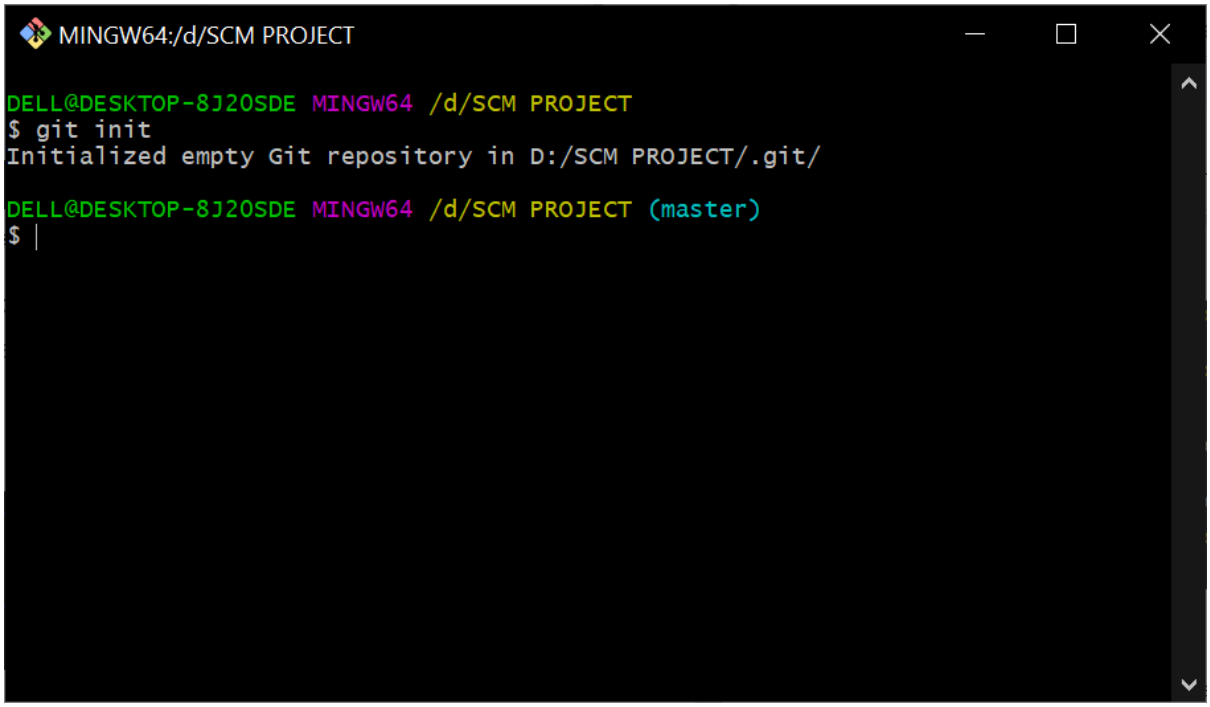
✧ Verify it from your email and you are all set to go.



Experiment No. 03

Aim: Program to Generate logs

✧ 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.



✧ 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 →

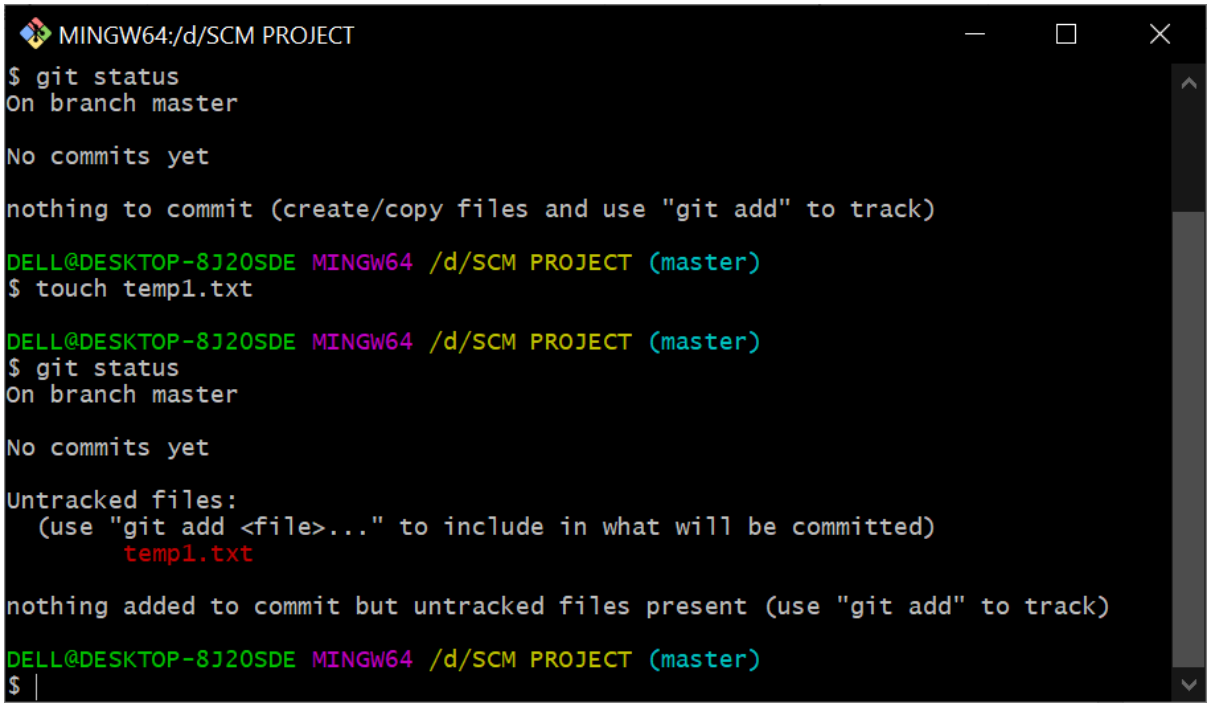
"git config --global user.name"

"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.

Now, we have to create some files in the repository. Suppose we created index.html Now type git status:



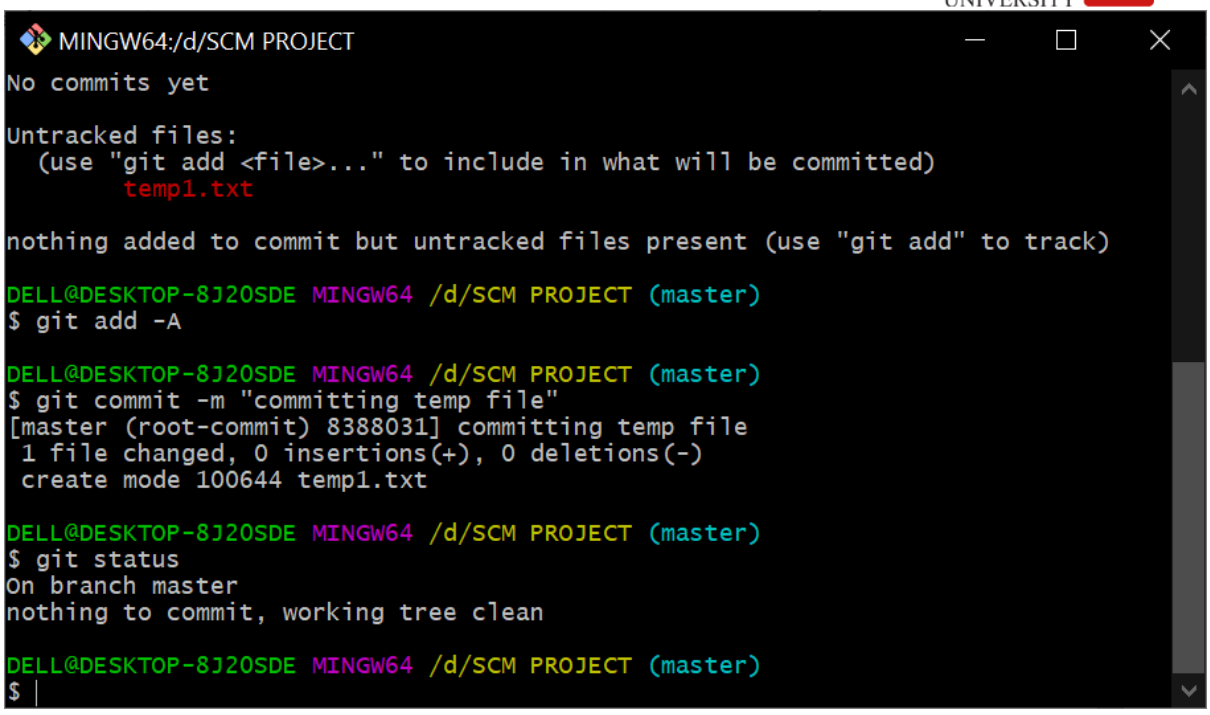
You can see that *index.html* is in red colour that means it is an untracked file.

Now firstly add the file in staging area and then commit the file.

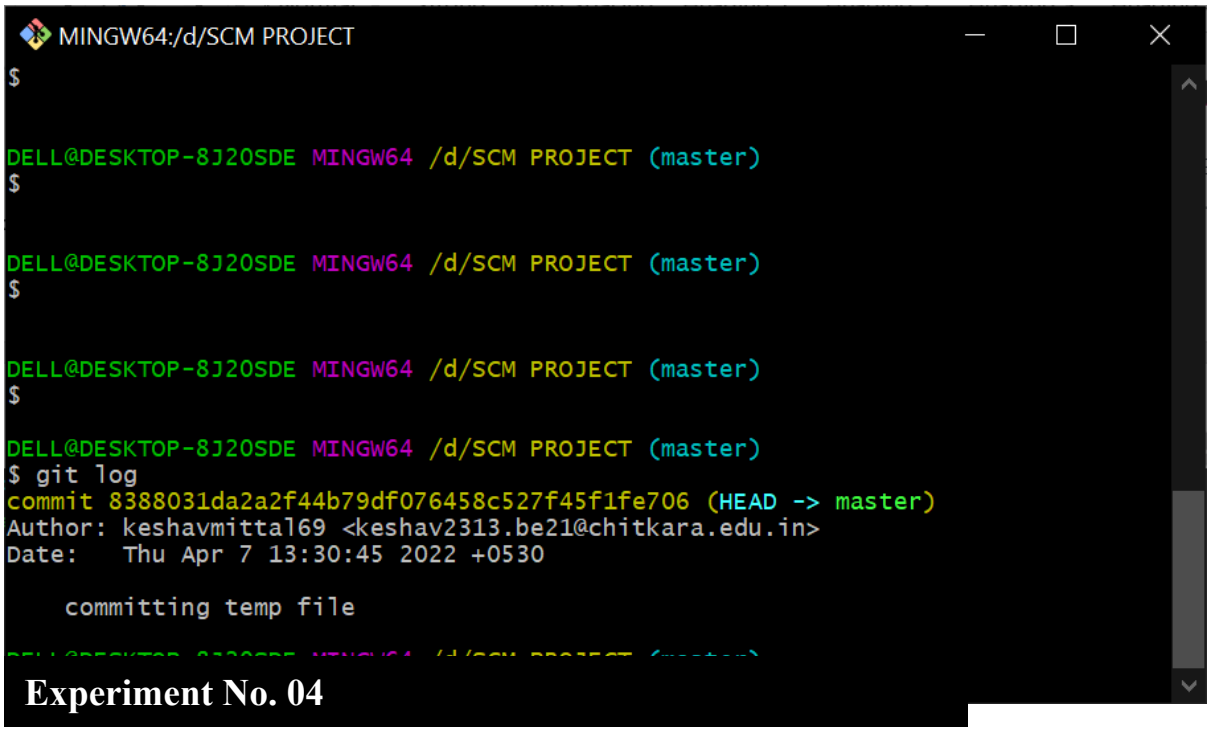
For this, use command →

git add -A [For add all the files in staging area.]

git commit -m "write any message" [For commit the file]



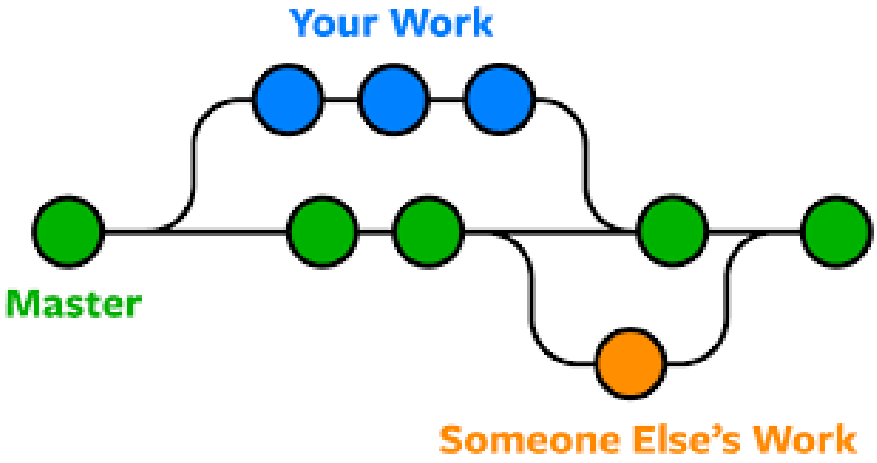
✧ **git log:** *The git log command displays a record of the commits in a Git repository. By default, the git log command displays a commit hash, the commit message, and other commit metadata.*



Experiment No. 04

Aim: Create and visualize branches

✧ **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 main branch.



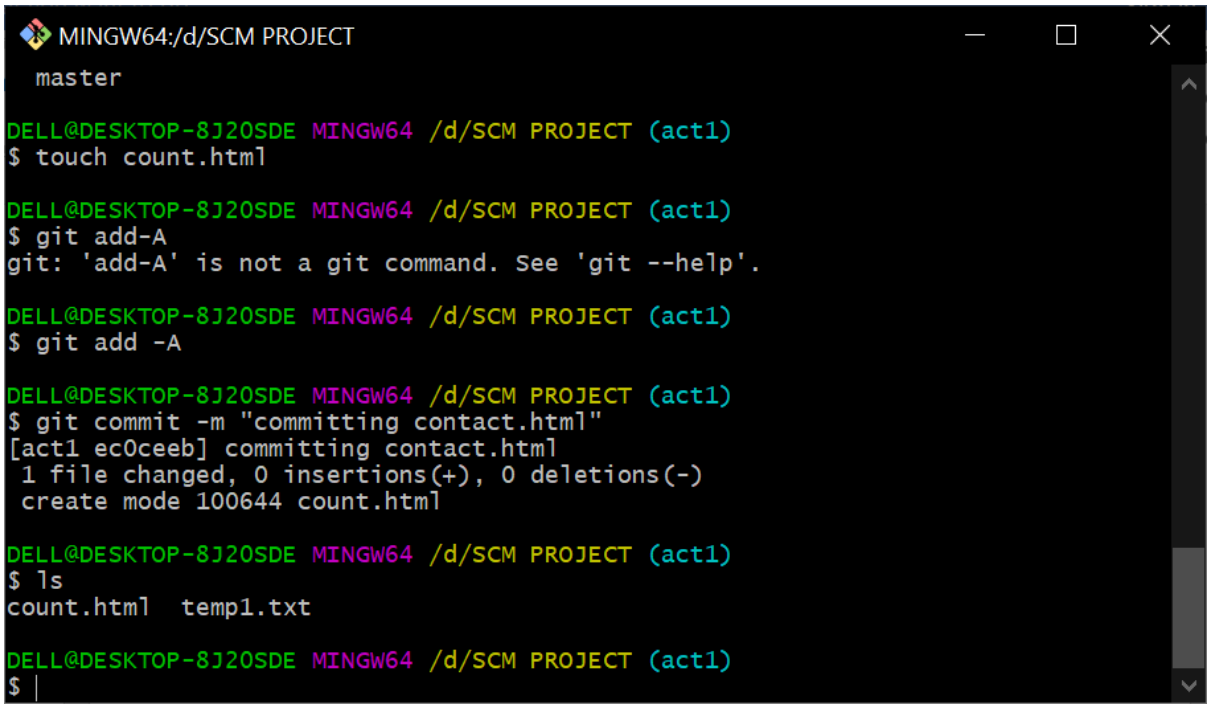
Let us see the command of it:
Firstly, add a new branch, let us suppose the branch name is activity1.
For this use command →

- **git branch name** [adding new branch]
- **git branch** [use to see the branch's names]
- **git checkout *branch name*** [use to switch to the given branch]

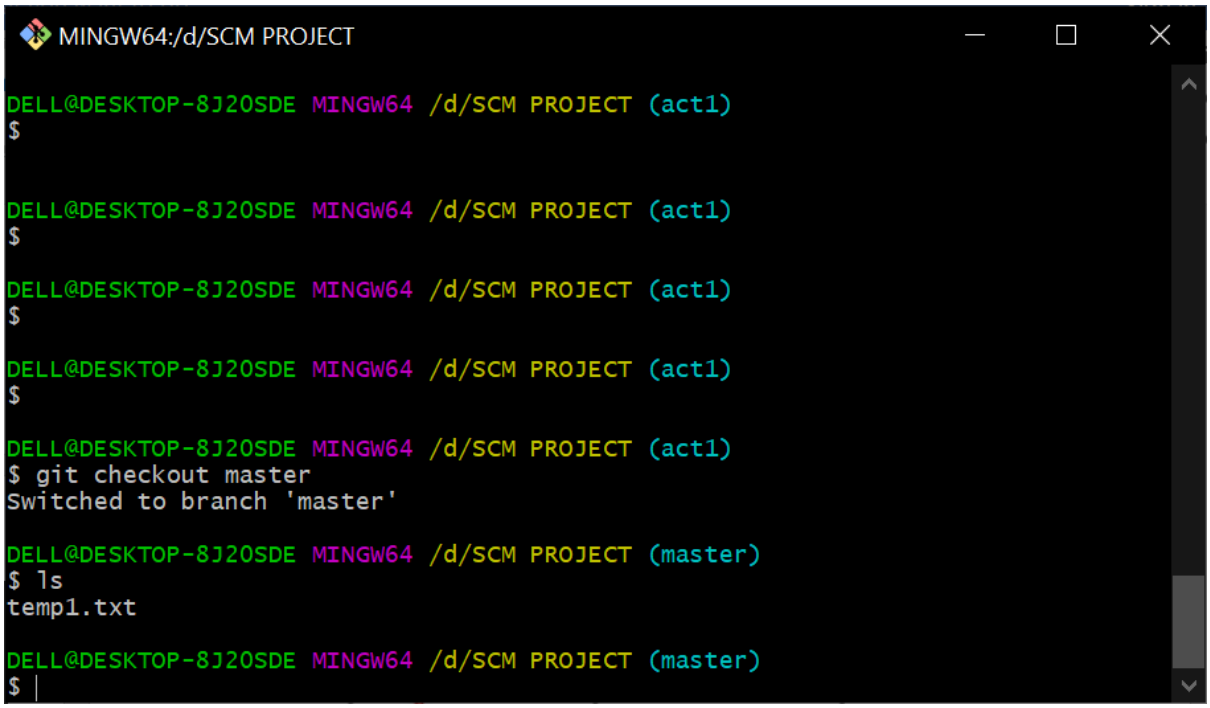
```
MINGW64:/d/SCM PROJECT
commit 8388031da2a2f44b79df076458c527f45f1fe706 (HEAD -> master)
Author: keshavmittal69 <keshav2313.be21@chitkara.edu.in>
Date:   Thu Apr 7 13:30:45 2022 +0530

    committing temp file

DELL@DESKTOP-8J20SDE MINGW64 /d/SCM PROJECT (master)
$ git branch
* master

DELL@DESKTOP-8J20SDE MINGW64 /d/SCM PROJECT (master)
$ git branch act1

DELL@DESKTOP-8J20SDE MINGW64 /d/SCM PROJECT (master)
$ git checkout act1
Switched to branch 'act1'

DELL@DESKTOP-8J20SDE MINGW64 /d/SCM PROJECT (act1)
$ git branch
* act1
  master

DELL@DESKTOP-8J20SDE MINGW64 /d/SCM PROJECT (act1)
$
```

In this you can see that firstly 'git branch' shows only one branch in green colour but when we add a new branch using 'git branch act1', it shows 2 branches but the green colour and star is on master. So, we have to switch to act1 by using 'git checkout act1'. If we use 'git branch', now you can see that the green colour and star is on act1. It means you are in activity1 branch and all the data of master branch is also on act1 branch. Use "ls" to see the files.

Now add a new file in activity1 branch, do some changes in file and commit the file.



If we switched to master branch, 'contact.html' file is not there. But he file is in activity1 branch.



- To add these files in master branch, we have to do merging. For this firstly switch to master branch and then use command →

  git merge branchname [use to merge branch]

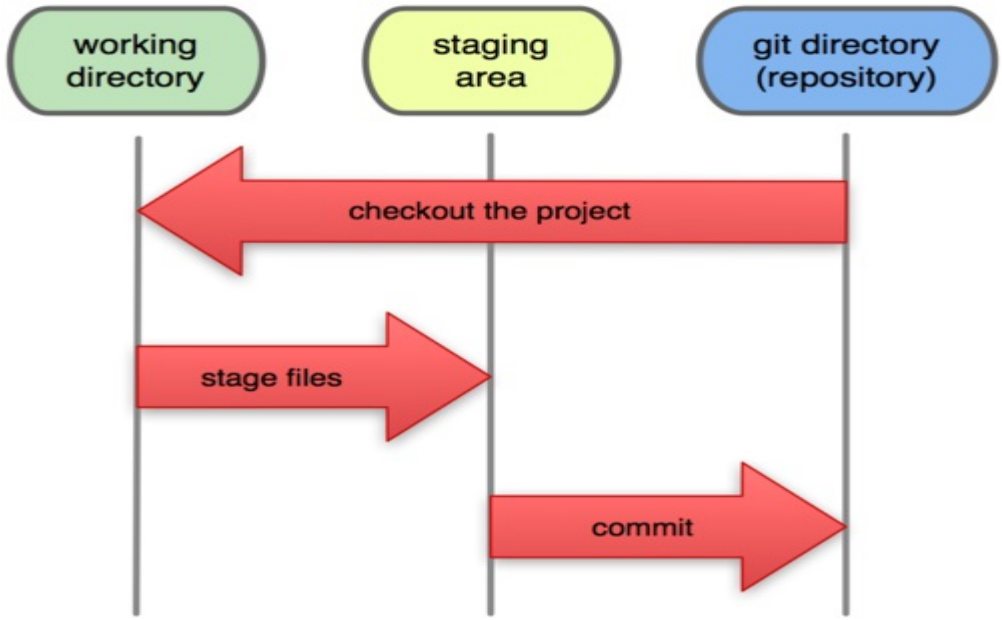
```
MINGW64:/d/SCM PROJECT
DELL@DESKTOP-8J2OSDE MINGW64 /d/SCM PROJECT (act1)
$

DELL@DESKTOP-8J2OSDE MINGW64 /d/SCM PROJECT (act1)
$

DELL@DESKTOP-8J2OSDE MINGW64 /d/SCM PROJECT (act1)
$ git checkout master
Switched to branch 'master'

DELL@DESKTOP-8J2OSDE MINGW64 /d/SCM PROJECT (master)
$ ls
temp1.txt

DELL@DESKTOP-8J2OSDE MINGW64 /d/SCM PROJECT (master)
$ git log
commit 8388031da2a2f44b79df076458c527f45f1fe706 (HEAD -> master)
Author: keshavmittal69 <keshav2313.be21@chitkara.edu.in>
Date:   Thu Apr 7 13:30:45 2022 +0530

    committing temp file

DELL@DESKTOP-8J2OSDE MINGW64 /d/SCM PROJECT (master)
$
```

Experiment No. 05

Aim: Git lifecycle description

Stages in GIT Life Cycle:

Now let's understand the three-stage architecture of Git:



- **Working Directory:** This is the directory that we've initialized, and here all the changes are made to commit on GitHub.

- **Staging Area:** This is where we first put out code or files of the working repository. The command that we use to stage code is, "git add --a", "git add FileName" or "git add -A". In simple terms, staging means telling Git what files we want to commit (new untracked files, modified files, or deleted files).

- **Git directory(repository):** This is where all the commits are stored whenever we make a commit. We can revert to an older version of or project using the "git checkout" command from this directory.

Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **DCSE**

Submitted By:
Name Abhishek Kumar Gupta
2110990066
G01

Submitted To:
.

 # List of Programs #

Experiment No. 06

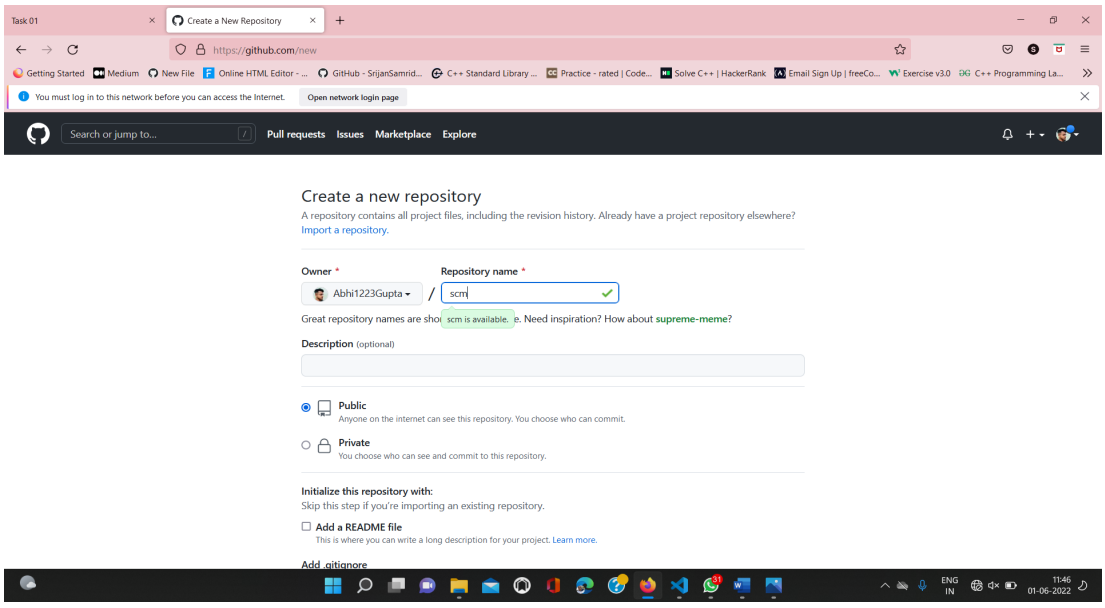
Aim: Add collaborators on Github Repo

Theory:

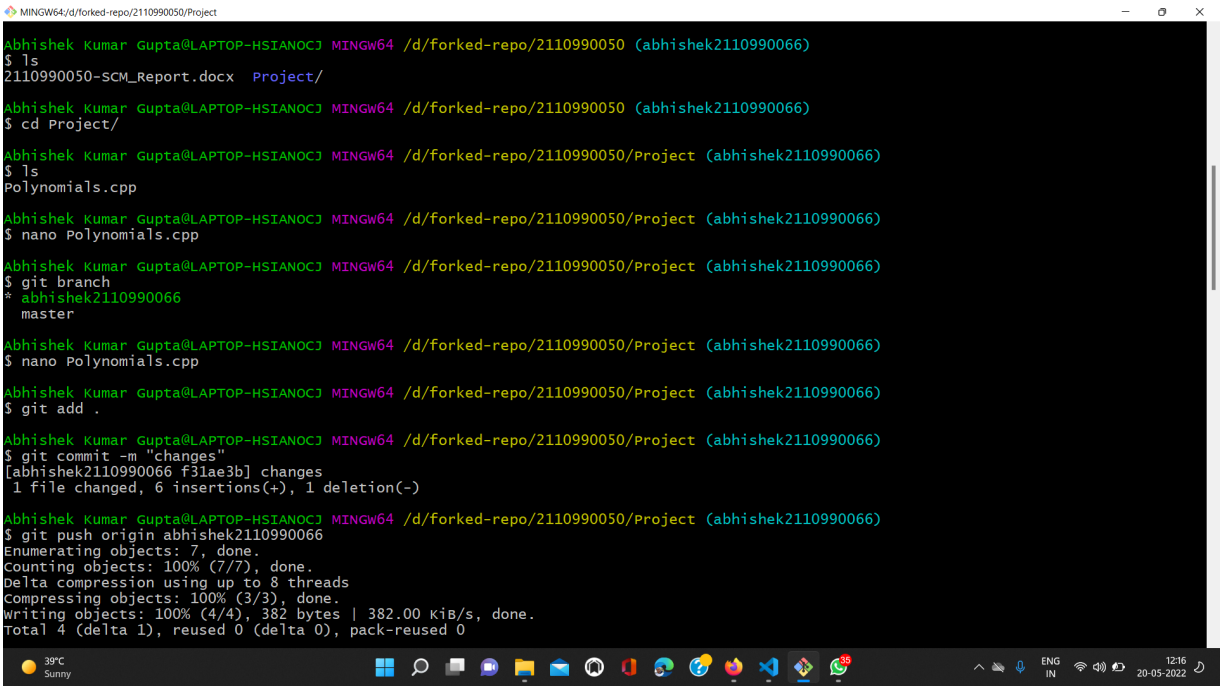
1. Create a New Repository.

   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, and data sets, as well as the revision history for all files in the repository. For more information, see "About repositories."

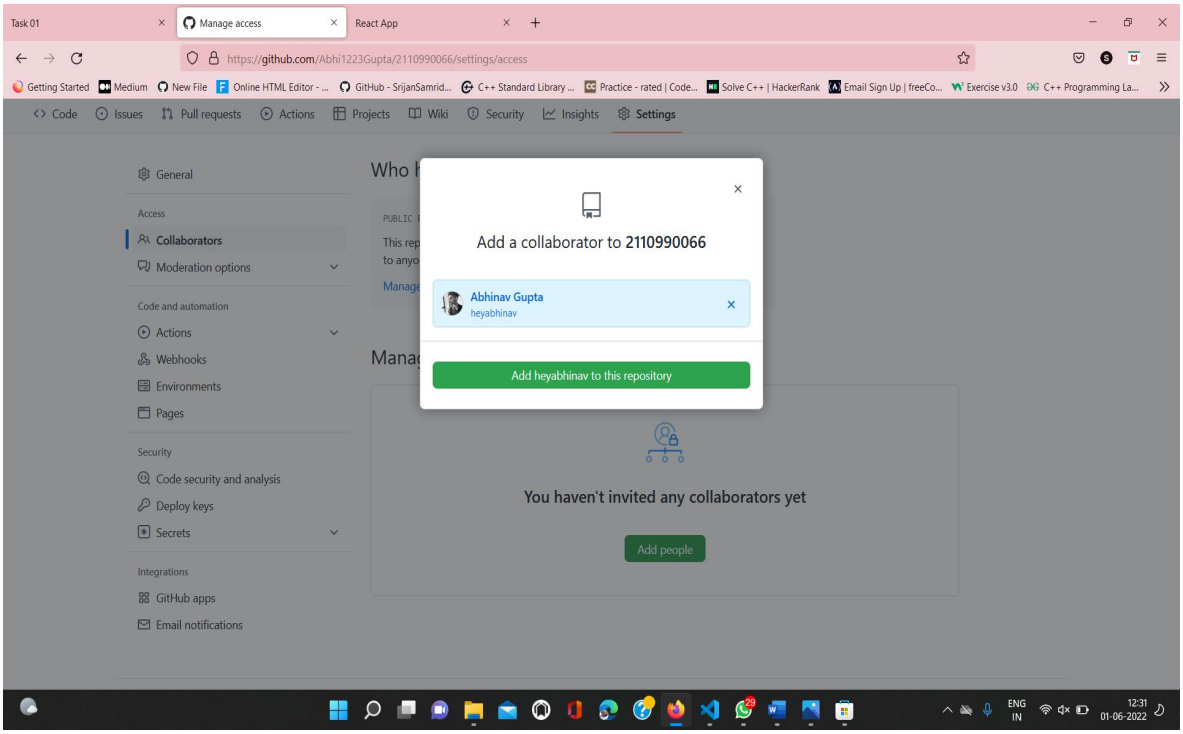
   When you create a new repository, you should initialize the repository with a README file to let people know about your project. For more information, see "Creating a new repository."



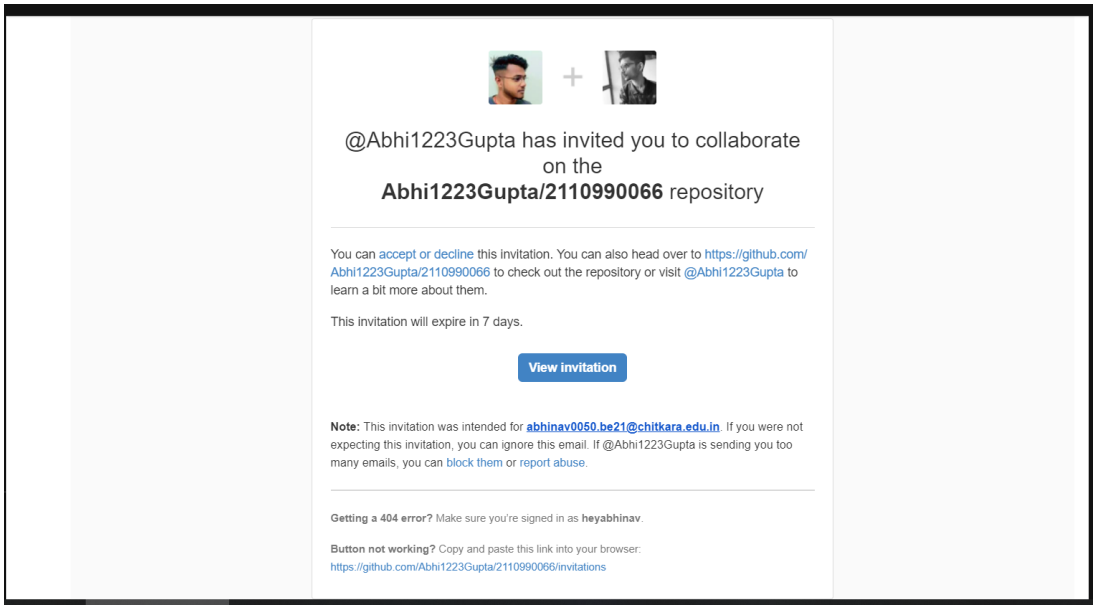
2. Now Copy the HTTP link of your repo and paste it it on your ‘Git CLI’, and merge the local repo in remote repo .


MINGW64:/d/forked-repo/2110990050/Project
Abhishek Kumar Gupta@LAPTOP-HSIANOCJ MINGW64 /d/forked-repo/2110990050 (abhishek2110990066)
$ ls
2110990050-SCM_Report.docx Project/
Abhishek Kumar Gupta@LAPTOP-HSIANOCJ MINGW64 /d/forked-repo/2110990050 (abhishek2110990066)
$ cd Project/
Abhishek Kumar Gupta@LAPTOP-HSIANOCJ MINGW64 /d/forked-repo/2110990050/Project (abhishek2110990066)
$ ls
Polynomials.cpp
Abhishek Kumar Gupta@LAPTOP-HSIANOCJ MINGW64 /d/forked-repo/2110990050/Project (abhishek2110990066)
$ nano Polynomials.cpp
Abhishek Kumar Gupta@LAPTOP-HSIANOCJ MINGW64 /d/forked-repo/2110990050/Project (abhishek2110990066)
$ git branch
* abhishek2110990066
  master
Abhishek Kumar Gupta@LAPTOP-HSIANOCJ MINGW64 /d/forked-repo/2110990050/Project (abhishek2110990066)
$ nano Polynomials.cpp
Abhishek Kumar Gupta@LAPTOP-HSIANOCJ MINGW64 /d/forked-repo/2110990050/Project (abhishek2110990066)
$ git add .
Abhishek Kumar Gupta@LAPTOP-HSIANOCJ MINGW64 /d/forked-repo/2110990050/Project (abhishek2110990066)
$ git commit -m "changes"
[abhishek2110990066 f31ae3b] changes
 1 file changed, 6 insertions(+), 1 deletion(-)
Abhishek Kumar Gupta@LAPTOP-HSIANOCJ MINGW64 /d/forked-repo/2110990050/Project (abhishek2110990066)
$ git push origin abhishek2110990066
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 382 bytes | 382.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
39°C
Sunny
ENG
IN
12:16
20-05-2022

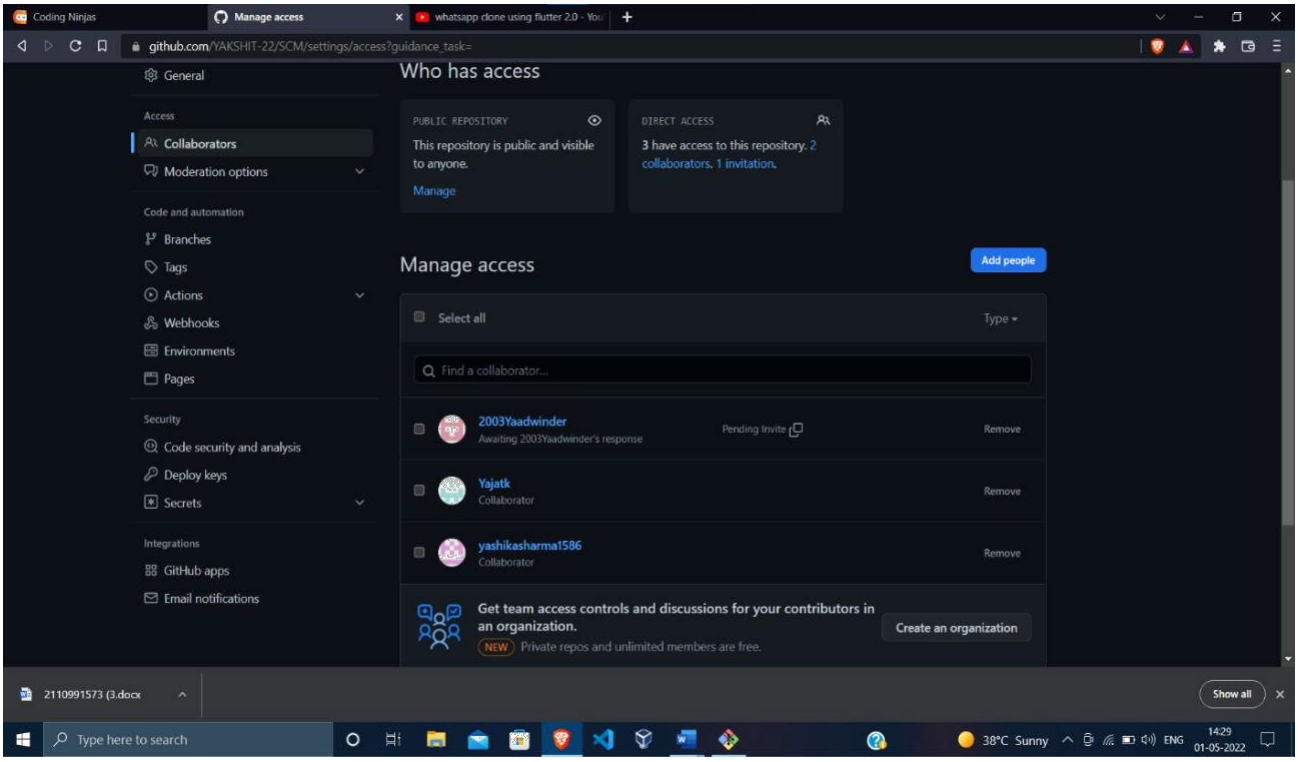
3. Go to Collaborators in Repo Setting, Add the username or email of Collaborator you want to add in your Repo.



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



5. New Collaborator has now access to the YAKSHIT-22 Repo.


Coding Ninjas
Manage access
whatsapp clone using flutter 2.0 - You
github.com/YAKSHIT-22/SCM/settings/access?guidance_task=
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Actions
Webhooks
Environments
Pages
Security
Code security and analysis
Deploy keys
Secrets
Integrations
GitHub apps
Email notifications
Who has access
PUBLIC REPOSITORY
This repository is public and visible to anyone.
Manage
DIRECT ACCESS
3 have access to this repository. 2 collaborators. 1 invitation.
Manage access
Add people
Select all
Type
Find a collaborator...
2003Yaadwinder
Awaiting 2003Yaadwinder's response
Pending Invite
Remove
Yajatk
Collaborator
Remove
yashikasharma1586
Collaborator
Remove
Get team access controls and discussions for your contributors in an organization.
NEW
Private repos and unlimited members are free.
Create an organization
2110991573 (3.docx
Show all
Type here to search
38°C Sunny
ENG
14:29
01-05-2022

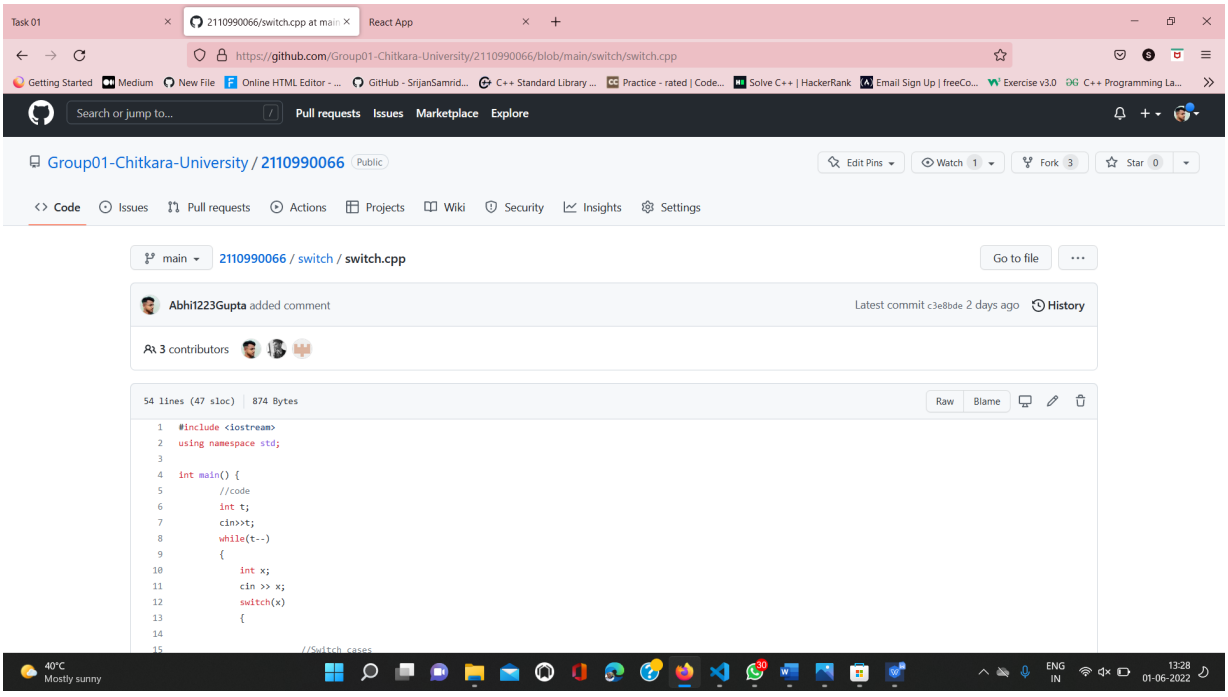
Experiment No. 07

Aim: Fork and Commit

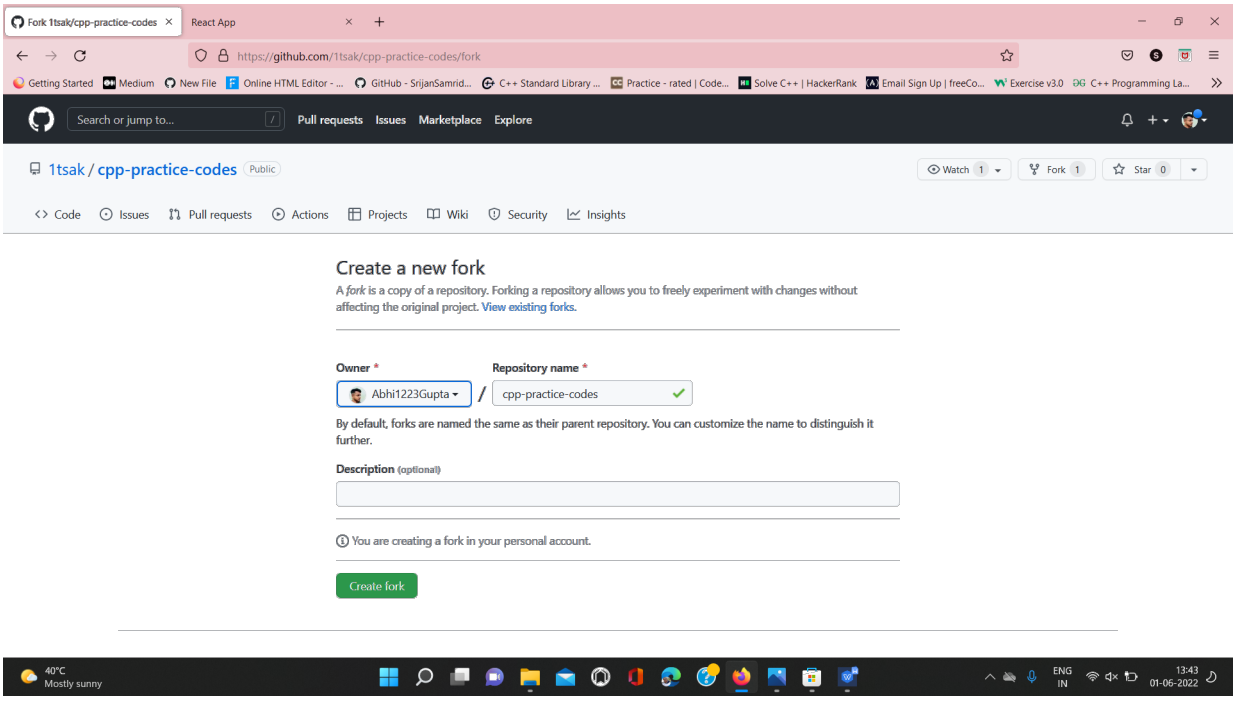
Theory:

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

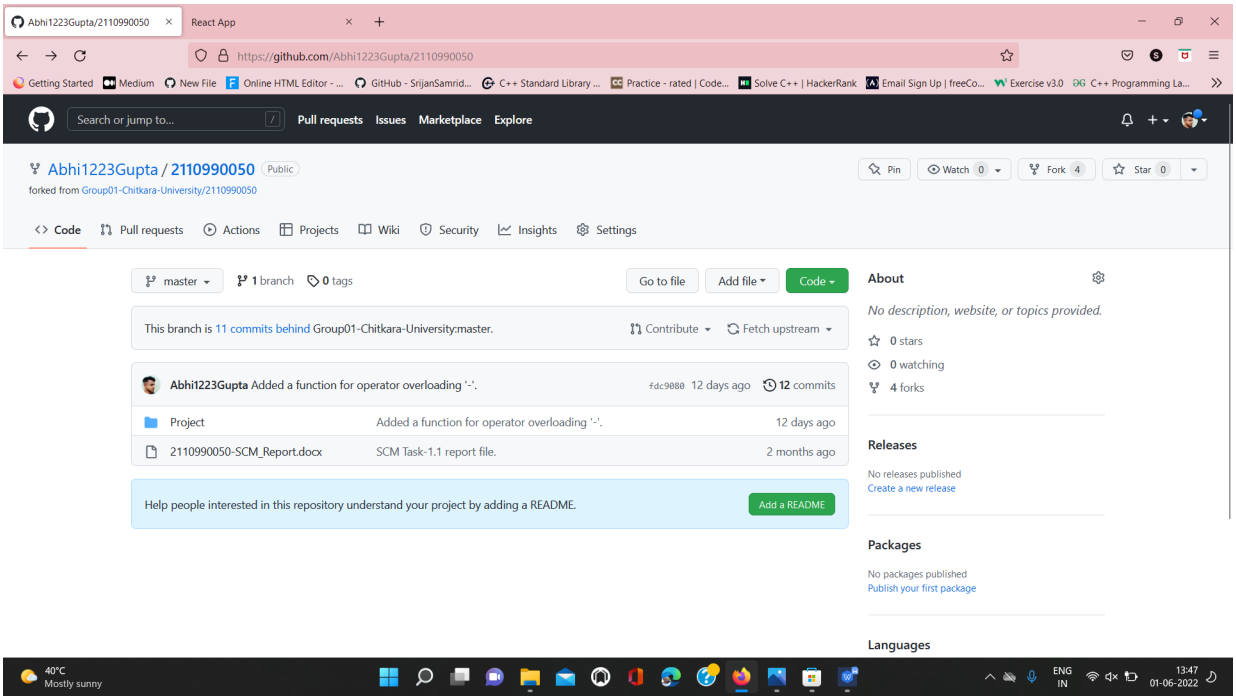
1. To fork a repository first thing you need to do is, sign in to your GitHub account and then you came to the repository you want to fork, so here for demo purpose am using **Yajatk/Demo** repository.



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

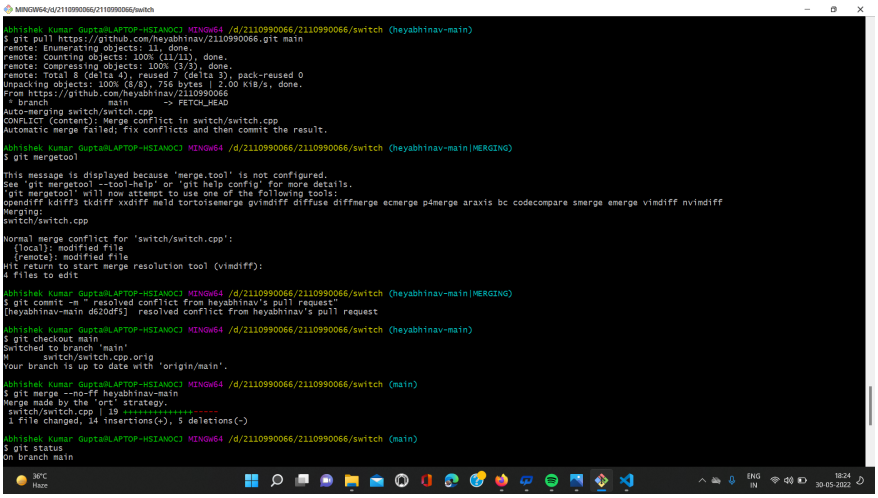


3. Now you will have a copy of the repo you have forked from other user. Now you can do any modification you want without making changes to main source code.

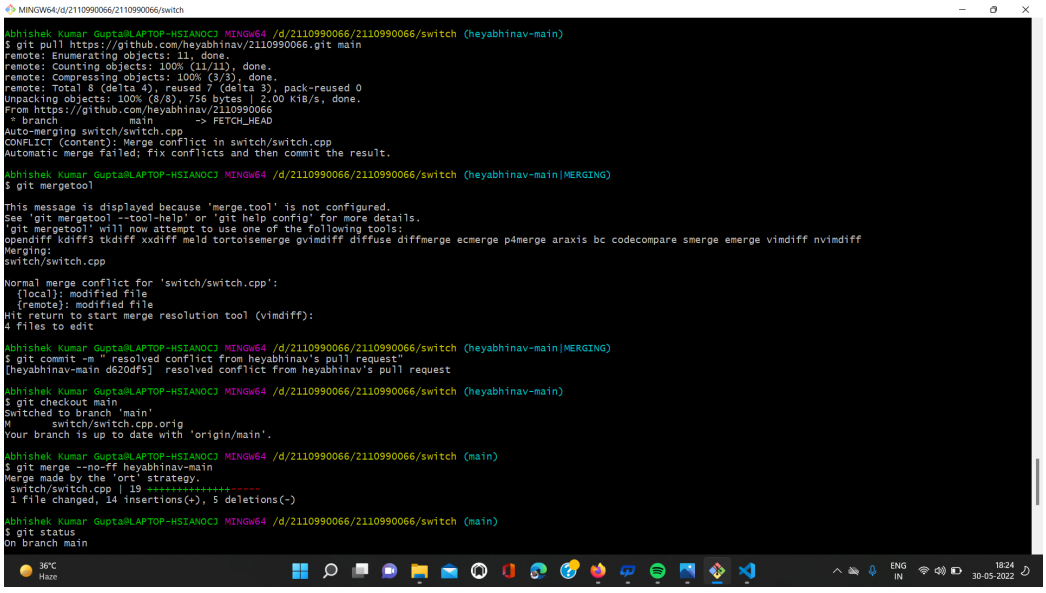


4. Now type https://github.com/Group01-Chitkara-University/2110990066_on CLI.

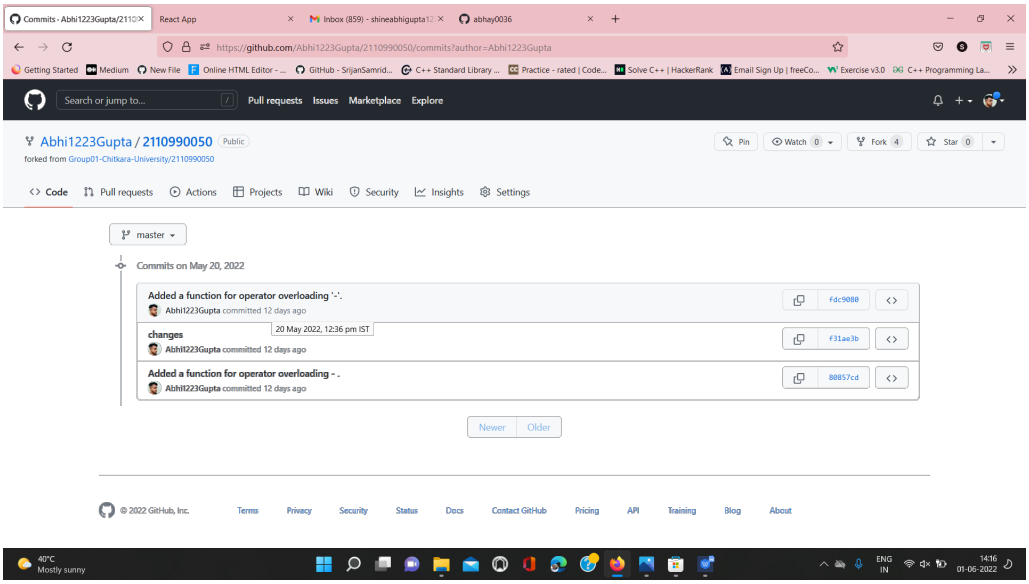
   Git pull <url> --> This command is used to fetch the remote repo or to clone the repo.



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.


Commits - Abhi1223Gupta/2110
React App
Inbox (859) - shineabhigupta12
abhay0036
https://github.com/Abhi1223Gupta/2110990050/commits?author=Abhi1223Gupta
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Abhi1223Gupta / 2110990050
Public
forked from Group01-Chitkara-University/2110990050
Pin
Watch 0
Fork 4
Star 0
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
Commits on May 20, 2022
Added a function for operator overloading '-'.
Abhi1223Gupta committed 12 days ago
fdc9080
changes
20 May 2022, 12:36 pm IST
Abhi1223Gupta committed 12 days ago
f31ae3b
Added a function for operator overloading - .
Abhi1223Gupta committed 12 days ago
80857cd
Newer
Older
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About
40°C
Mostly sunny
ENG
IN
14:16
01-06-2022

Experiment No. 08

Aim: Merge and Resolve Conflicts

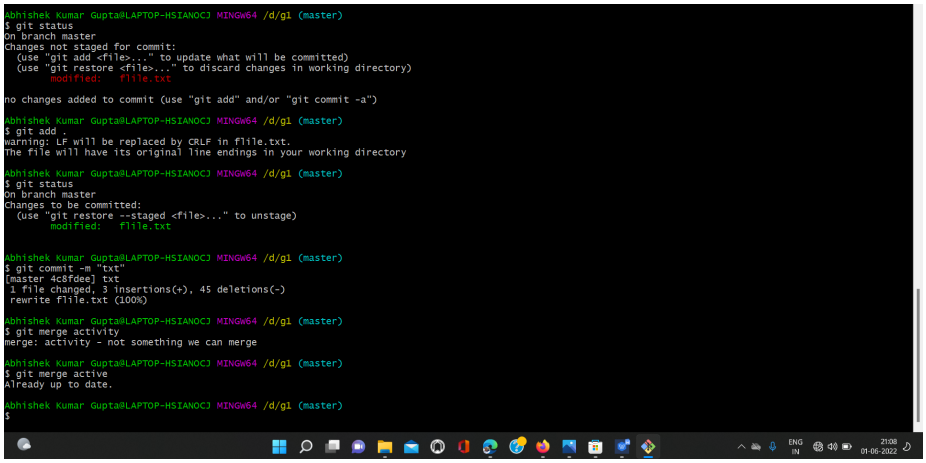
Theory:

1. Do changes in master branch and commit those change. And checkout to different branch and again do changes and commit it. Now checkout to master branch and merge that branch in master.

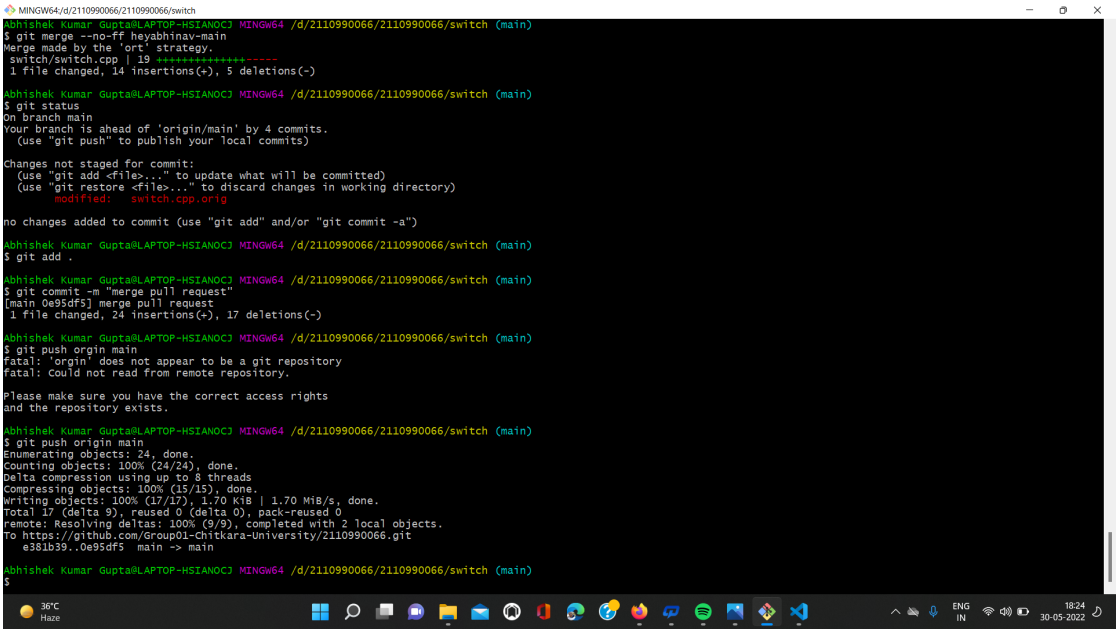
COMMIT IN MASTER BRANCH

COMMIT IN ACTIVITY BRANCH

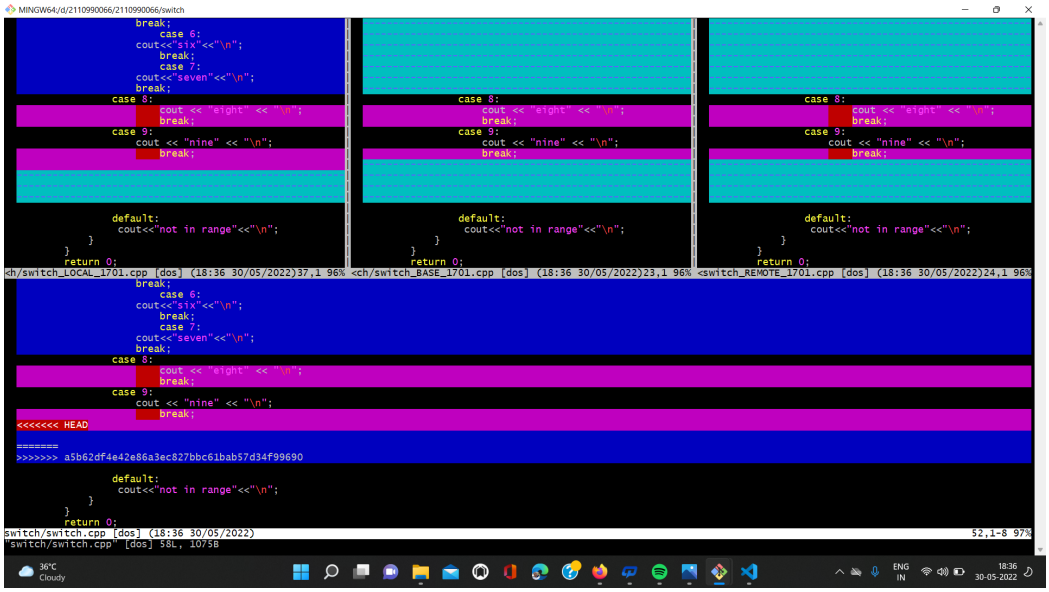
2. Now try to merge it will give Conflicts Error.



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



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


MINGW64:/d/2110990066/2110990066/switch
<ch/switch_LOCAL_1701.cpp [dos] (18:36 30/05/2022)37,1 96%
<ch/switch_BASE_1701.cpp [dos] (18:36 30/05/2022)23,1 96%
<switch_REMOTE_1701.cpp [dos] (18:36 30/05/2022)24,1 96%
<<<<<<< HEAD
=======
>>>>>>> a5b62df4e42e86a3ec827bbc61bab57d34f99690
switch/switch.cpp [dos] (18:36 30/05/2022) 52,1-8 97%
"switch/switch.cpp" [dos] 58L, 1075B
36°C
Cloudy
ENG
IN
18:36
30-05-2022

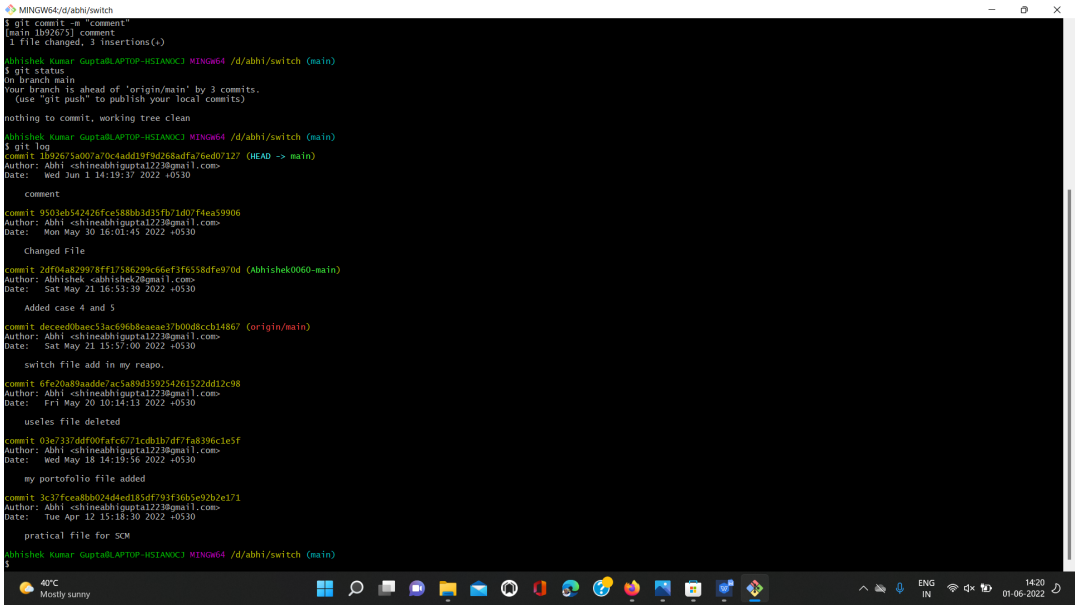
Experiment No. 09

Aim: Reset and Revert

Theory:

Git-revert – Revert some existing commits.

1. On GitBash CLI, Type command “git revert <Commit id>”. It revert the changes that done before Commit.



git revert HEAD~3 :-

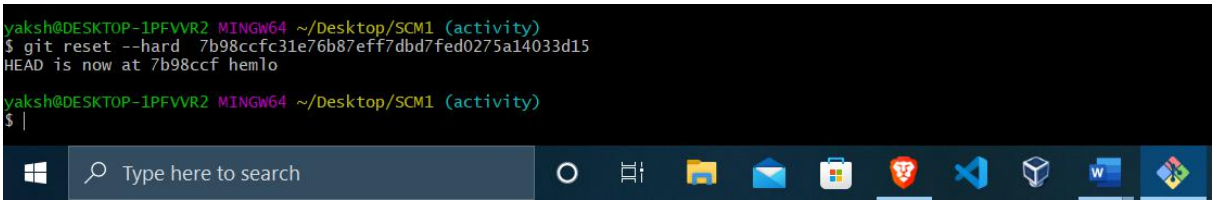
Revert the changes specified by the fourth last commit in HEAD and create a new commit with the reverted changes

Git-reset - Reset current HEAD to the specified state

2. At a surface level, git reset is similar in behavior 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 res

A Project report

on

"Task 2"

with

Source Code Management

(CS181)

Submitted by

Name :-Abhishek Kumar Gupta Roll No. :-2110990066

Team Member 1 Name :-Abhinav Gupta Roll No. :- 2110990050

Team Member 2 Name :- Abhay Kumar Roll No. :-2110990035

Team Member 3 Name :-Abhishek Dhamija Roll No. :- 2110990060

Department of Computer Science & Engineering

Chitkara University Institute of Engineering and Technology, Punjab

Jan- June

(2021-22)

Institute/School: - Name	**Chitkara University Institute of Engineering and Technology**		
Department Name: -	**Department of Computer Science & Engineering**		
Programme Name: -	**Bachelor of Engineering (B.E.), Computer Science & Engineering**		
Course Name: -	**Source Code Management**	Session: -	**2021-22**
Course Code: -	**CS181**	Semester/Batch: -	**2nd/2021**
Vertical Name: -	**Beta**	Group No: -	G01-B
Course Coordinator:-	**Dr. Monit Kapoor**		

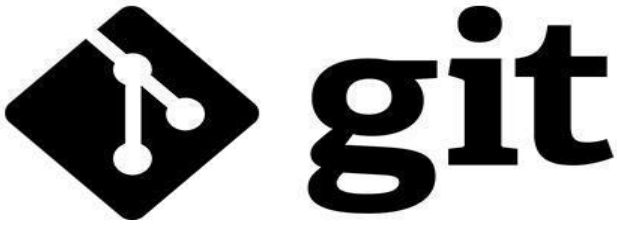
Faculty Name: -
Dr.Monit Kapoor

INDEX

Introduction

What is GIT and why is it used?

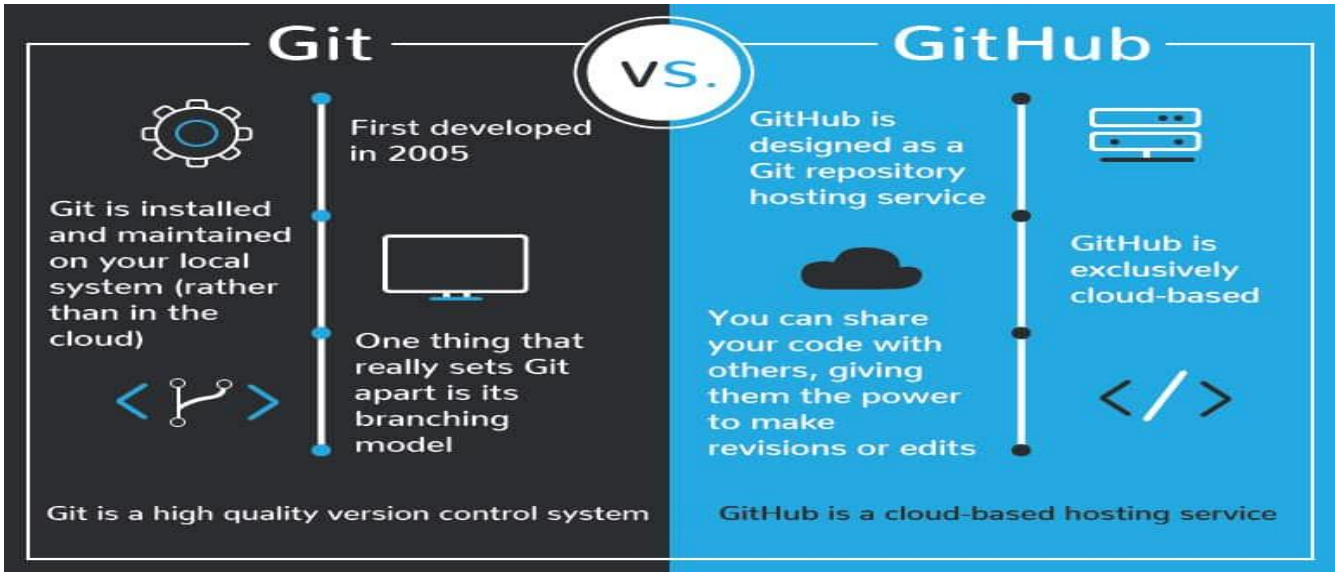
Git is a version control system that is widely used in the programming world. It is used for tracking changes in the source code during software development. It was developed in 2005 by Linus Torvalds, the creator of the Linux operating system kernel.
Git is a speedy and efficient distributed VCS tool that can handle projects of any size, from small to very large ones. Git provides cheap local branching, convenient staging areas, and multiple workflows. It is free, open-source software that lowers the cost because developers can use Git without paying money. It provides support for non-linear development. Git enables multiple developers or teams to work separately without having an impact on the work of others.
Git is an example of a distributed version control system (DVCS) (hence Distributed Version Control System).



What is GITHUB?

It is the world's largest open-source software developer community platform where the users upload their projects using the software Git.

What is the difference between GIT and GITHUB?



What is Repository?

A repository is a directory or storage space where your projects can live. Sometimes GitHub users shorten this to "repo." It can be local to a folder on your computer, or it can be a storage space on GitHub or another online host. You can keep code files, text files, image files, you name it, inside a repository.

What is Version Control System (VCS)?

A version control system is a tool that helps you manage "versions" of your code or changes to your code while working with a team over remote distances. Version control keeps track of every modification in a special kind of database that is accessible to the version control software. Version control software (VCS) helps you revert back to an older version just in case a bug or issue is introduced to the system or fixing a mistake without disrupting the work of other team members.

Types of VCS

1. Local Version Control System
2. Centralized Version Control System
3. Distributed Version Control System

I. **Local Version Control System:** Local Version Control System is located in your local machine. If the local machine crashes, it would not be possible to retrieve the files, and all the information will be lost. If anything happens to a single version, all the versions made after that will be lost.

AI. **Centralized Version Control System:** In the Centralized Version Control Systems, there will be a single central server that contains all the files related to the project, and many collaborators checkout files from this single server (you will only have a working copy). The problem with the Centralized

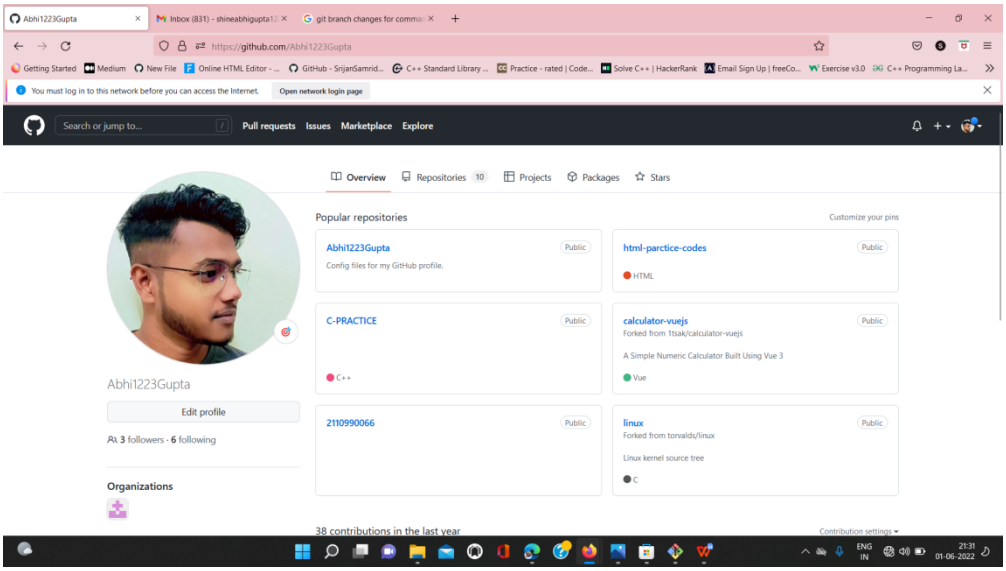
Version Control Systems is if the central server crashes, almost everything related to the project will be lost.

BI. **Distributed Version Control System:** In a distributed version control system, there will be one or more servers and many collaborators similar to the centralized system. But the difference is, not only do they check out the latest version, but each collaborator will have an exact copy of the main repository on their local machines. Each user has their own repository and a working copy. This is very useful because even if the server crashes we would not lose everything as several copies are residing in several other computers.

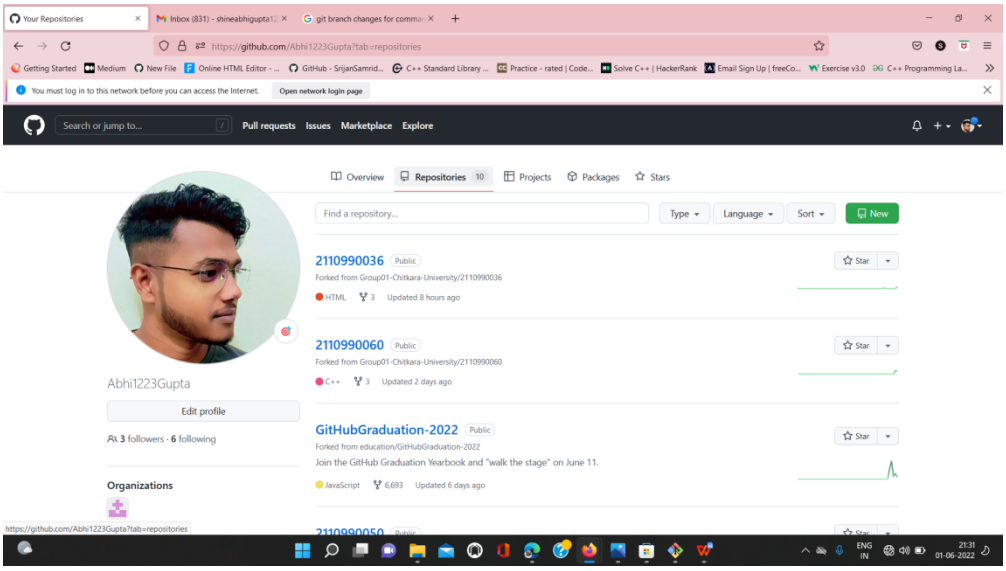
Experiment No. 01

Aim: Create a distributed Repository and add members in project team

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

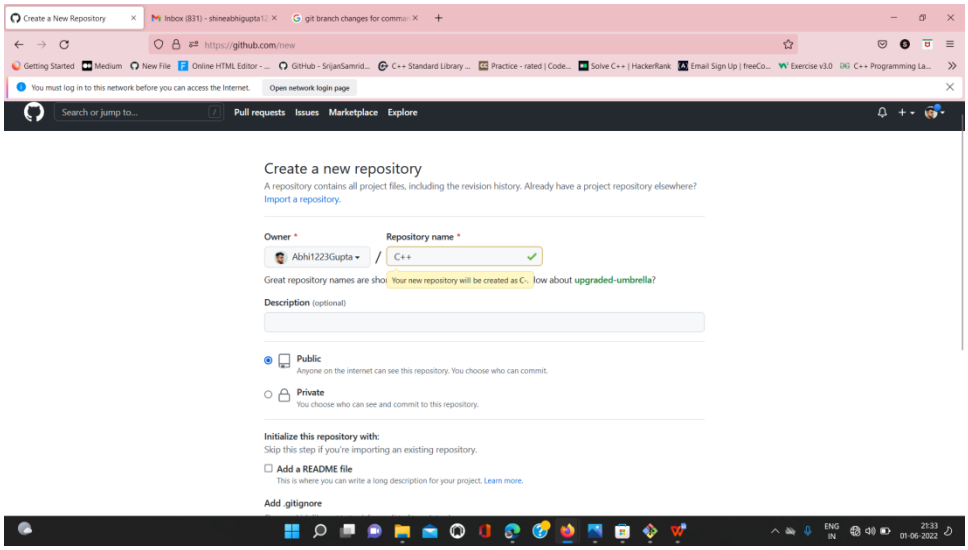


- Click on the 'New' button in the top right corner.

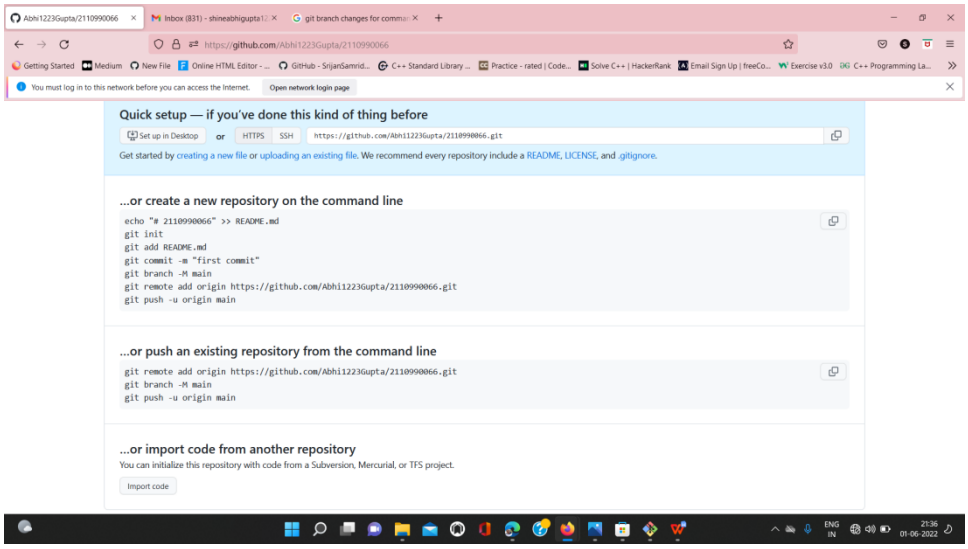


- Enter the Repository name and add the description of the repository.

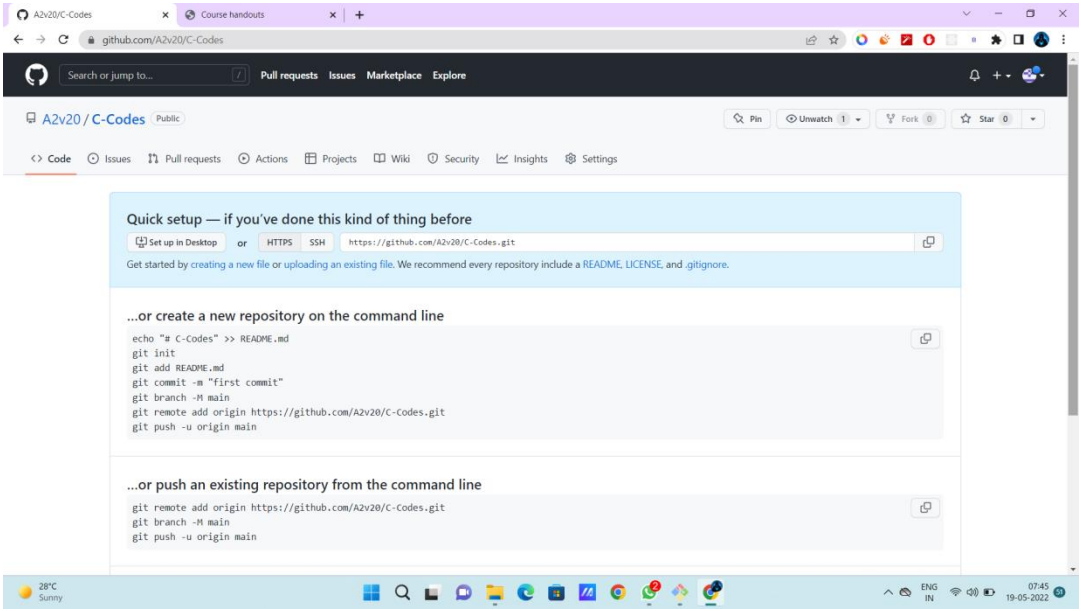
- Select if you want the repository to be public or private.



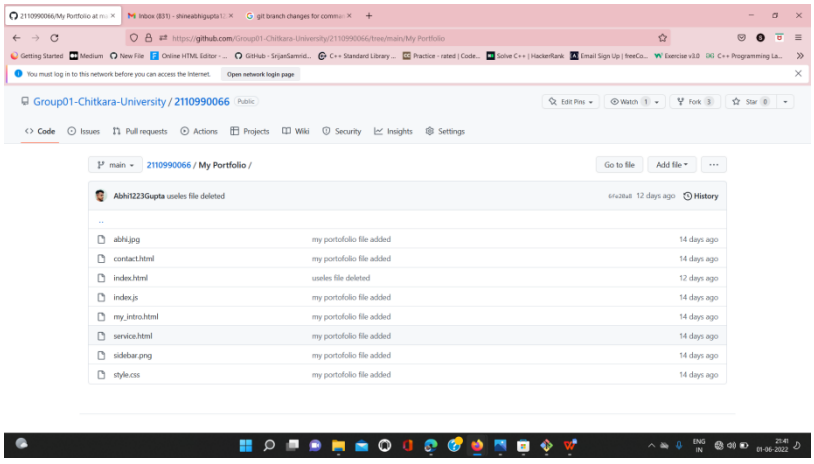
- If you want to import code from an existing repository select the import code option.



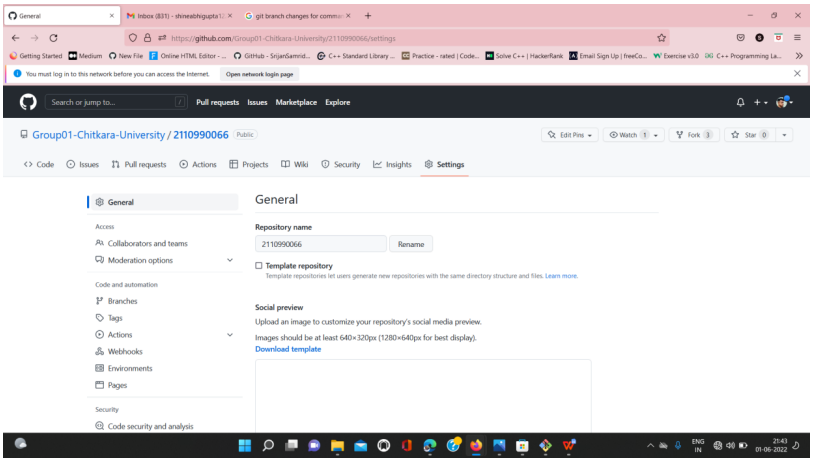
- To create a new file or upload an existing file into your repository select the option in the following box.



- Now, you have created your repository successfully.
- To add members to your repository open your repository and select settings option in the navigation bar.

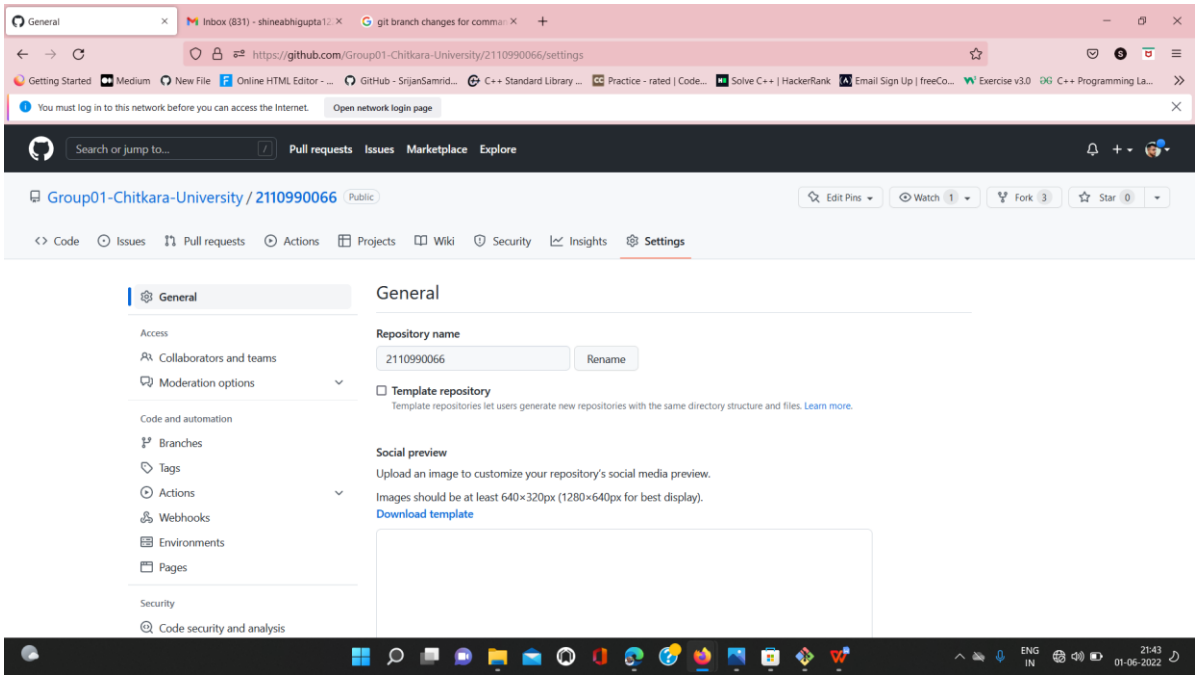


- Click on Collaborators option under the access tab.

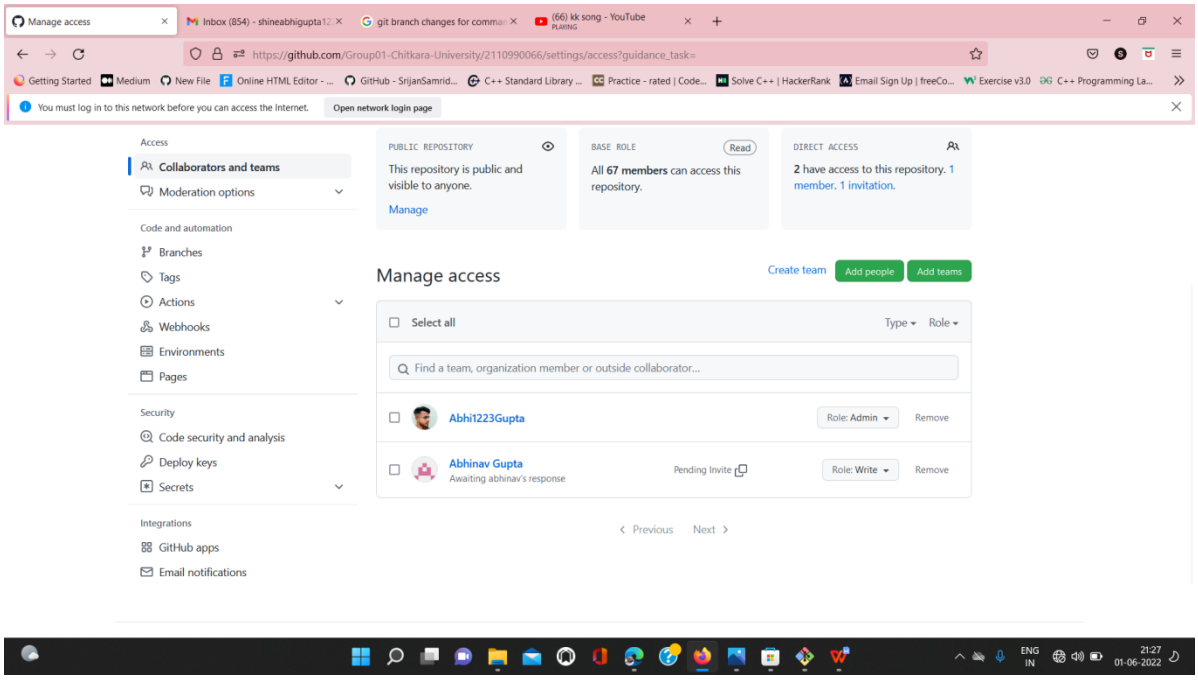


- After clicking on collaborators Github asks you to enter your password to confirm the access to the repository.

- After entering the password you can manage access and add/remove team members to your project.
- To add members click on the add people option and search the id of your respective team member.

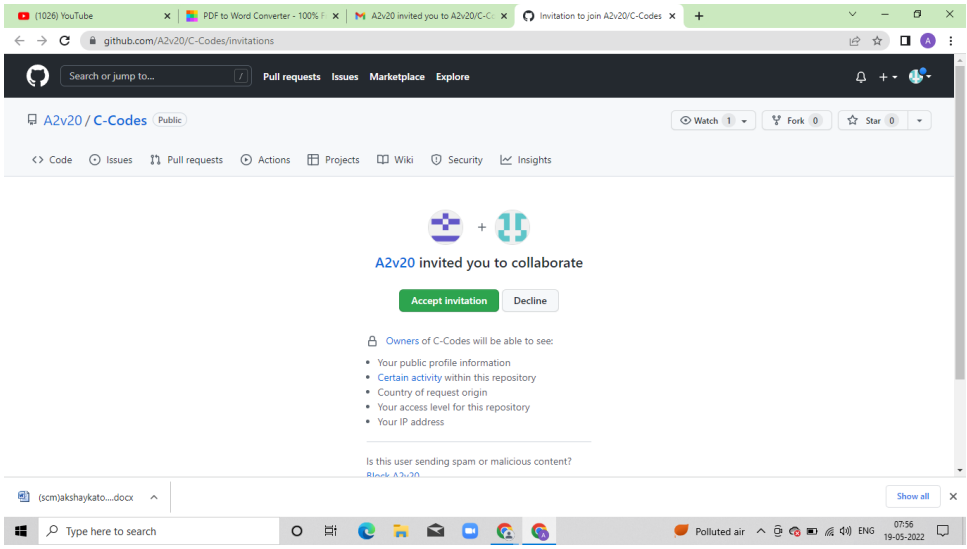


- To remove any member click on remove option available in the last column of member's respective row.



- To accept the invitation from your team member, open your email registered with Github.

- You will receive an invitation mail from the repository owner. Open the email and click on accept invitation.
- You will be redirected to Github where you can either select to accept or decline the invitation.



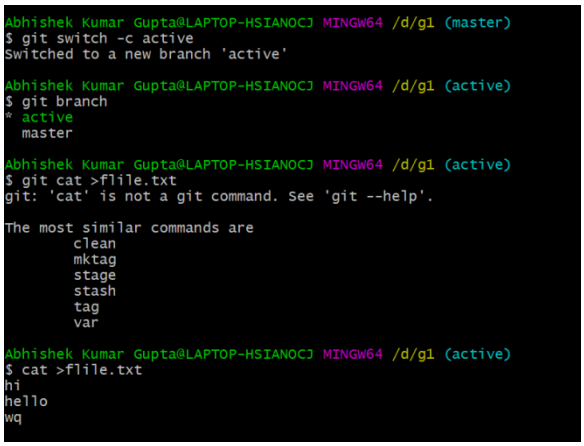
- You will be shown the option that you are now allowed to push.

- Now all members are ready to contribute to the project.

Experiment No. 02

Aim: Open And Close a Pull Request

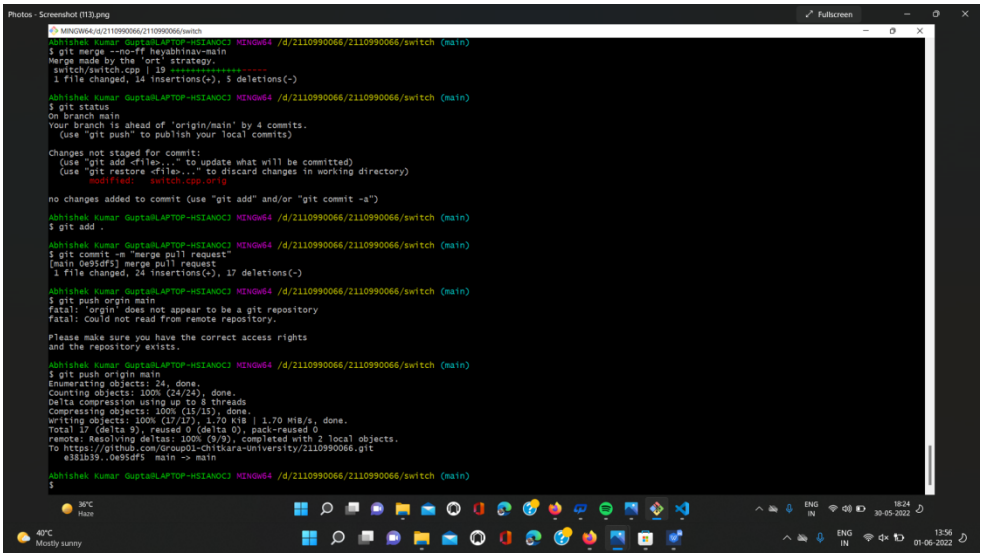
- To open a pull request we first have to make a new branch, by using git branch *branchname* option.
- After making new branch we add a file to the branch or make changes in the existing file.



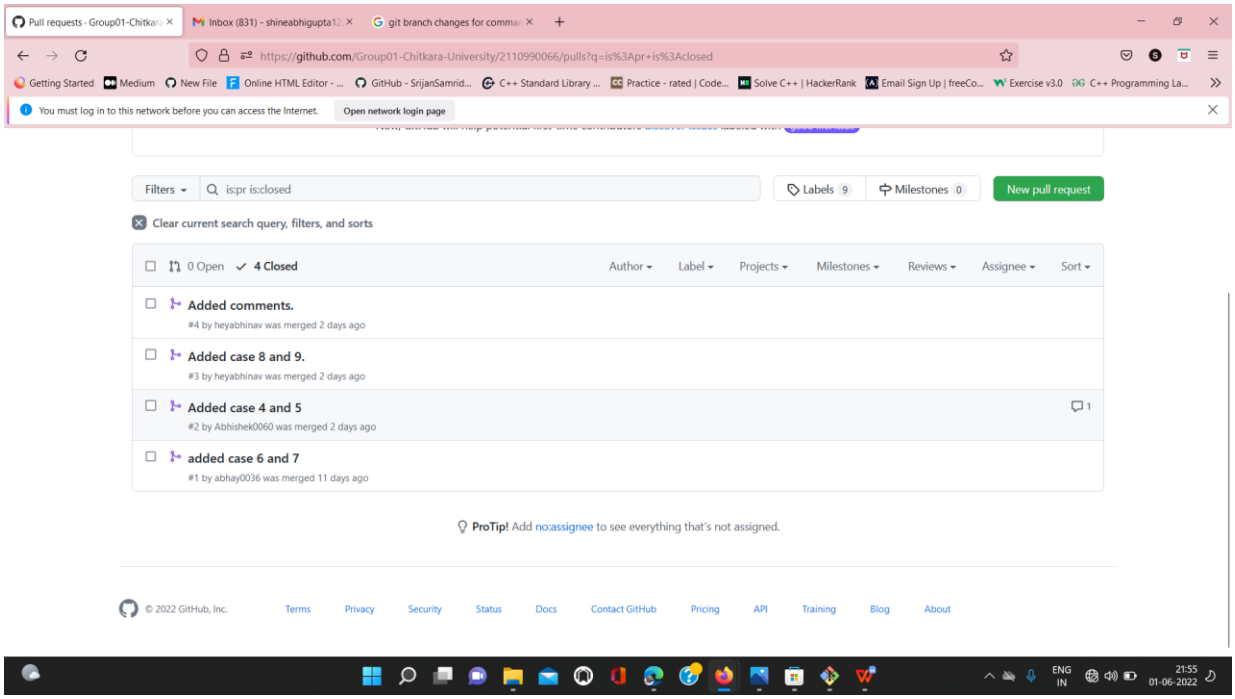
- Add and commit the changes to the local repository.

- Use git push origin *branchname* option to push the new branch to the main repository.

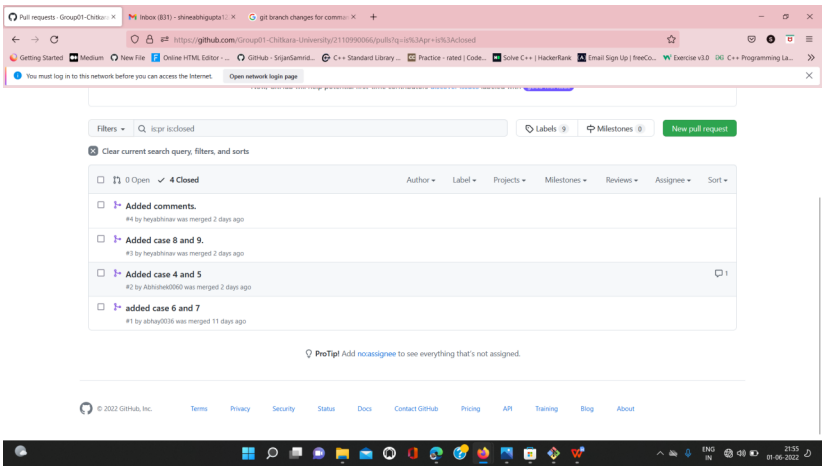
- After pushing new branch Github will either automatically ask you to



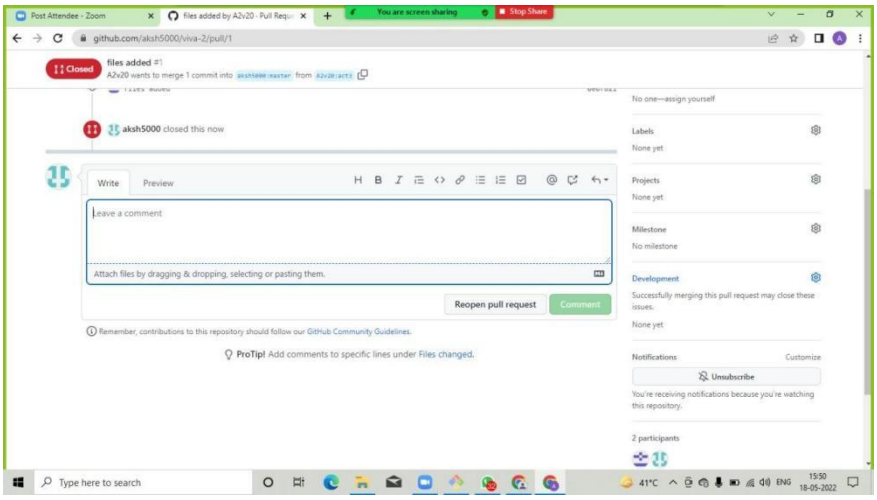
create a pull request or you can create your own pull request.



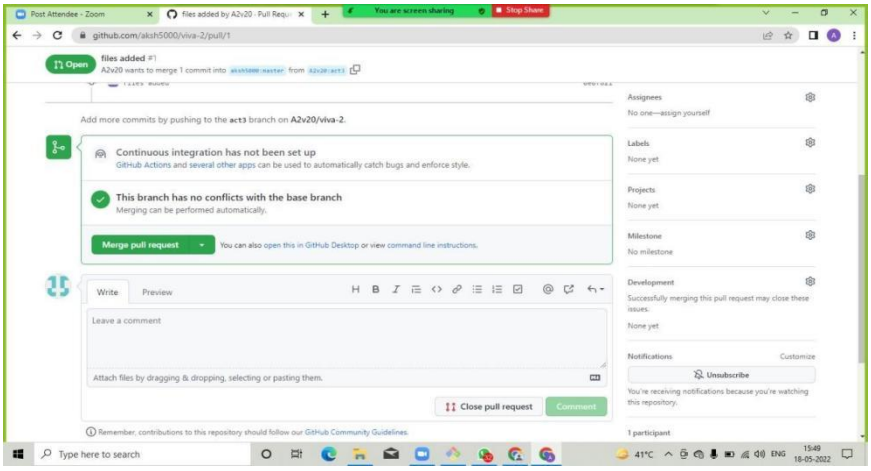
- To create your own pull request click on pull request option.



- Github will detect any conflicts and ask you to enter a description of your pull request.

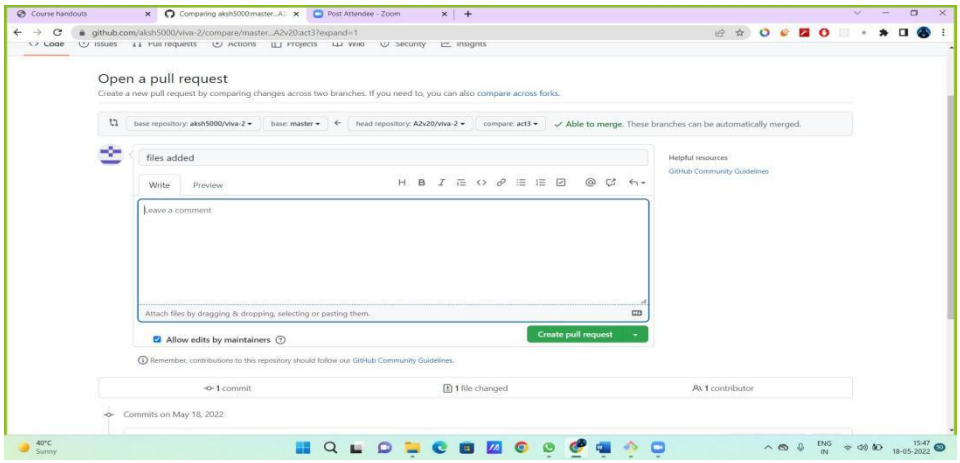


- After opening a pull request all the team members will be sent the request if they want to merge or close the request.

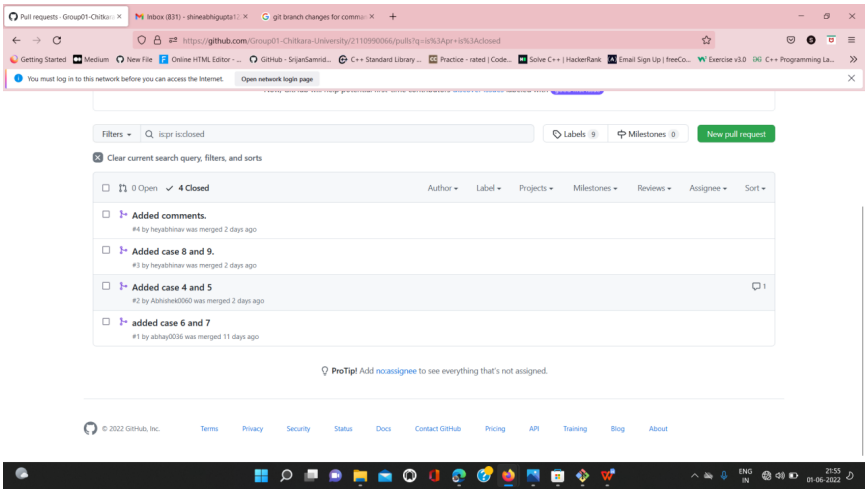


- If the team member chooses not to merge your pull request they will close you're the pull request.

- To close the pull request simply click on close pull request and add comment/ reason why you closed the pull request.



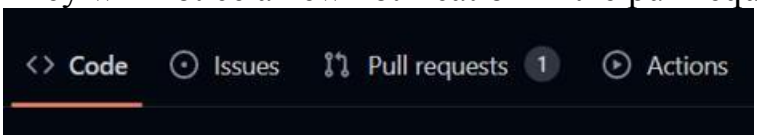
- You can see all the pull request generated and how they were dealt with by clicking on pull request option.

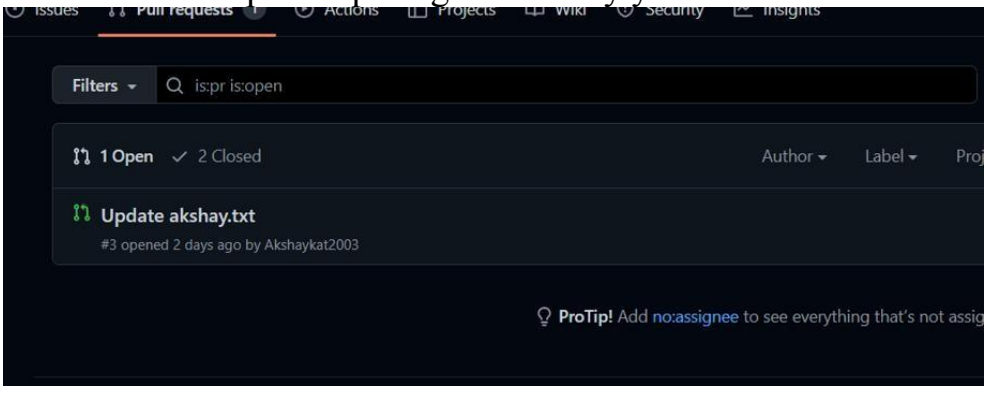


Experiment No. 03

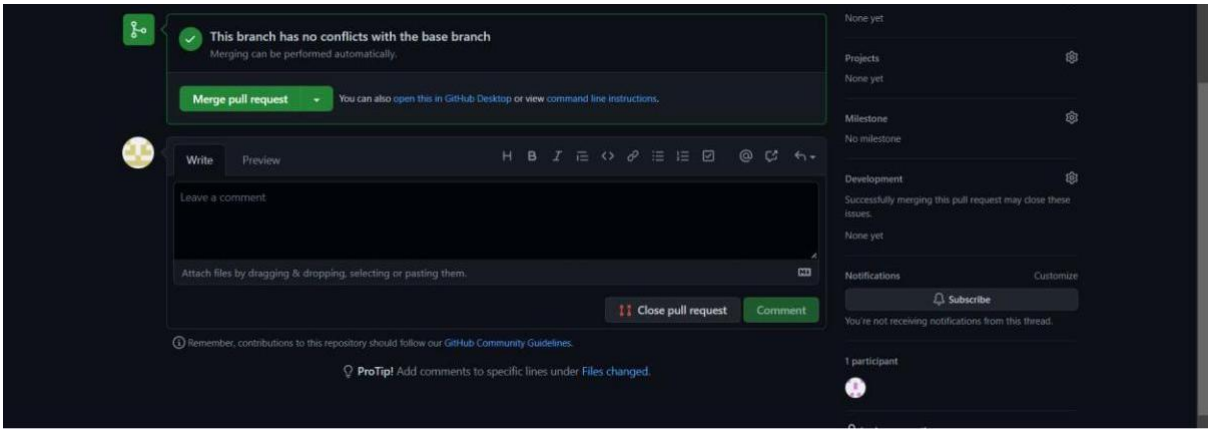
Aim: Create a pull request on a team member's repo and close pull requests generated by team members on own Repo as a maintainer

To create a pull request on a team member's repository and close requests by any other team members as a maintainer follow the procedure given below:-

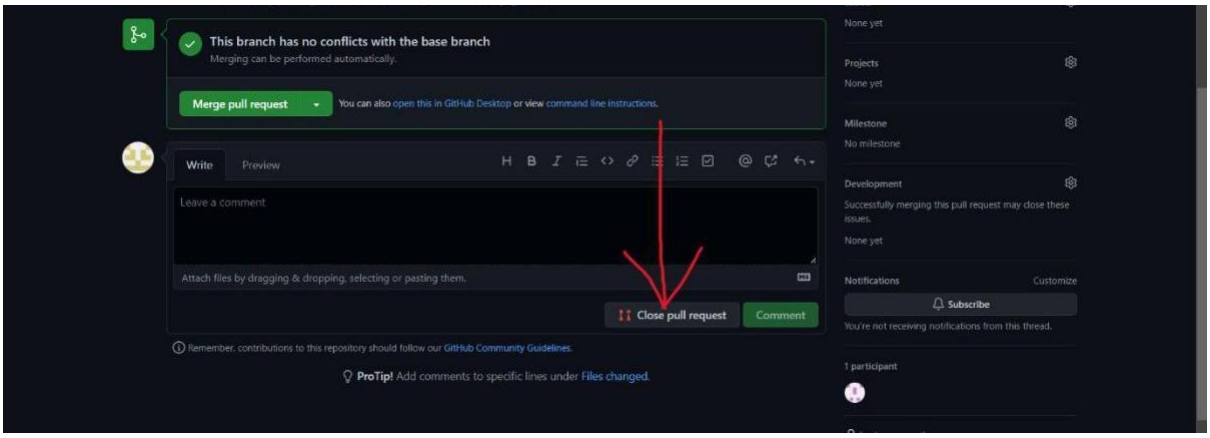
- Do the required changes in the repository, add and commit these changes in the local repository in a new branch.
- Push the modified branch using git push origin *branchname*.
- Open a pull request by following the procedure from the above experiment.
- The pull request will be created and will be 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.
- 

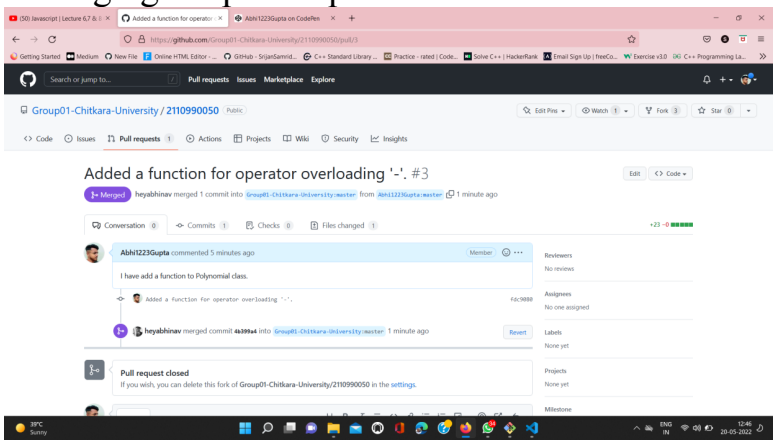
- Click on the pull request. Two option will be available, either to close the pull request or Merge the request with the main branch.
- By selecting the merge branch option the main branch will get updated for all the team members.



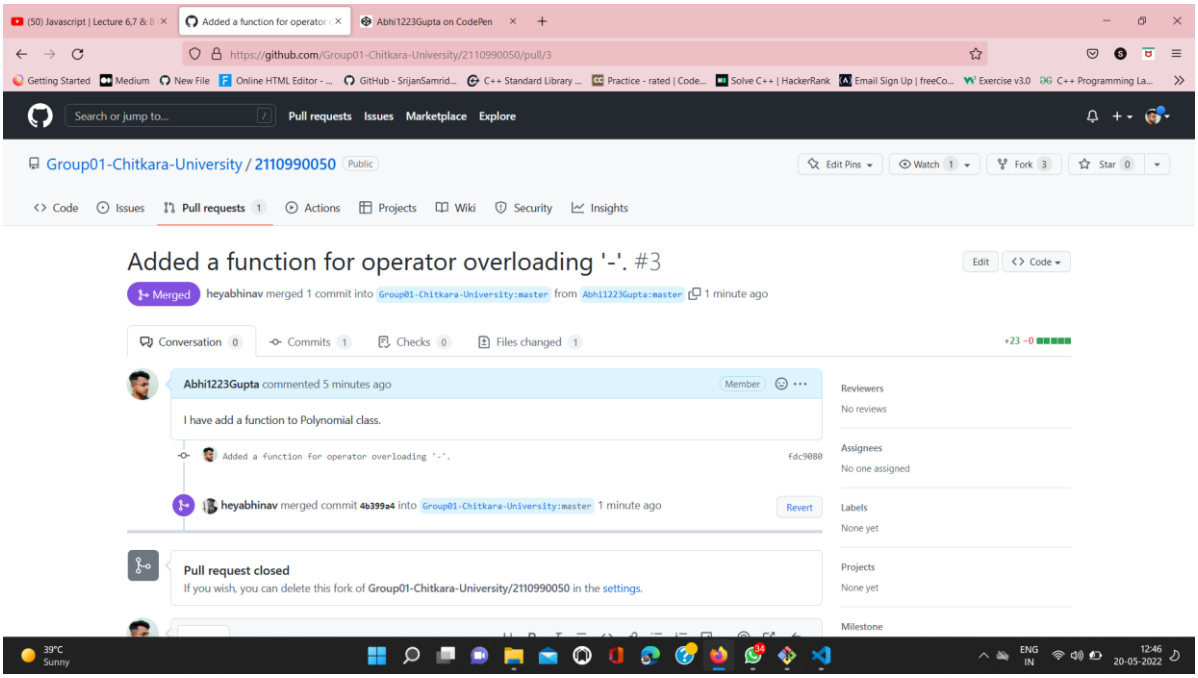
- By selecting close the pull request the pull request is not accepted and not merged with main branch.



- The process is similar to closing and merging the pull request by you. It simply includes an external party to execute.
- The result of merging the pull request is shown below.



- The result of closing the request is shown below.



- Thus, we conclude opening and closing of pull request. We also conclude merging of the pull request to the main branch.

Experiment No. 04

Aim: **Publish and print network graphs**

The network graph is one of the useful features for developers on GitHub. It is used to display the branch history of the entire repository network, including branches of the root repository and branches of forks that contain commits unique to the network.

A repository's graphs give you information on traffic, projects that depend on the repository, contributors and commits to the repository, and a repository's forks and network. If you maintain a repository, you can use this data to get a better understanding of who's using your repository and why they're using it.

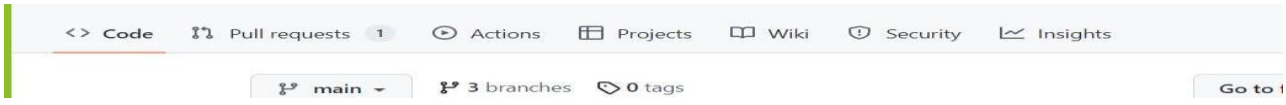
Some repository graphs are available only in public repositories with GitHub Free:

- Pulse
- Contributors
- Traffic
- Commits
- Code frequency
- Network

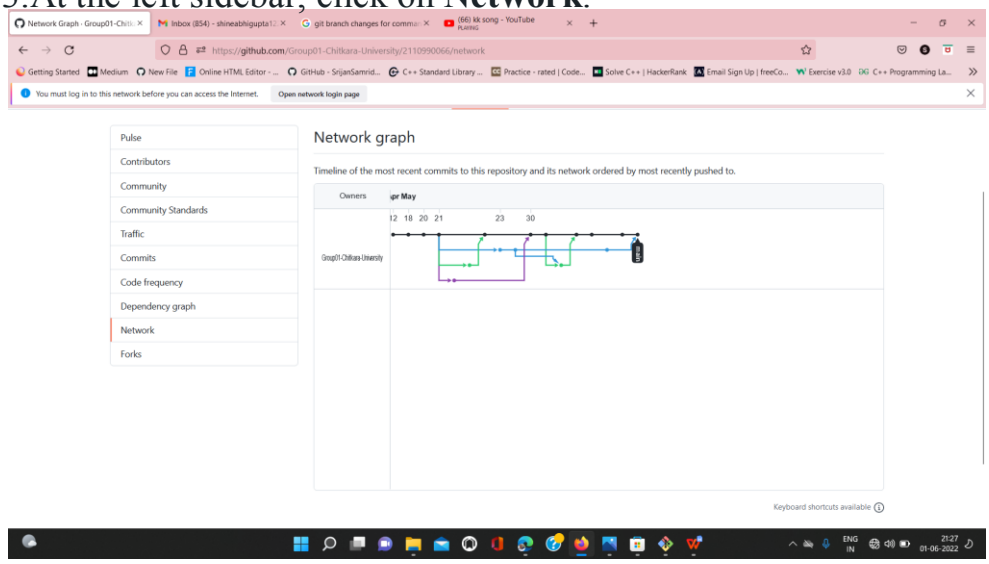
Steps to acess network graphs of respective repository

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

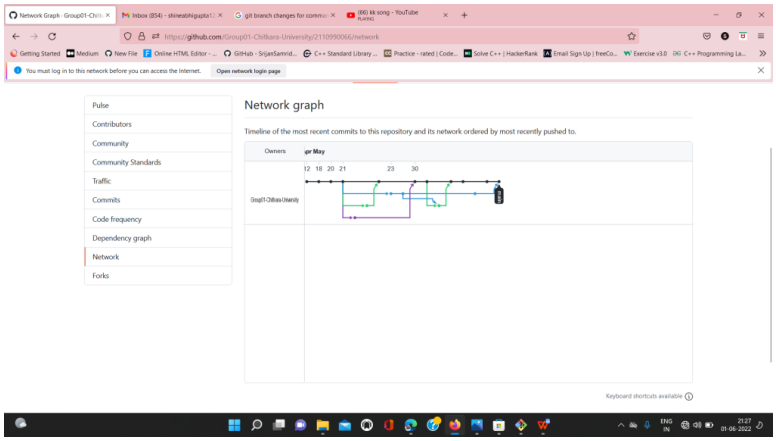
2.Under your repository name, click **Insights**.



3.At the left sidebar, click on **Network**.



You will get the network graph of your repository which displays the branch history of the entire repository network, including branches of the root repository and branches of forks that contain commits unique to the network.

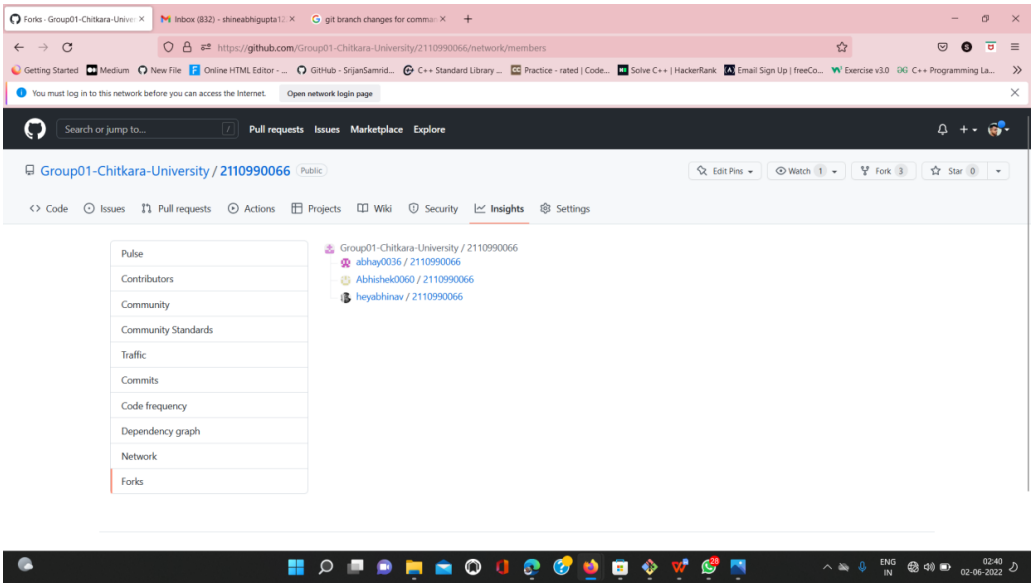


Listing the forks of a repository

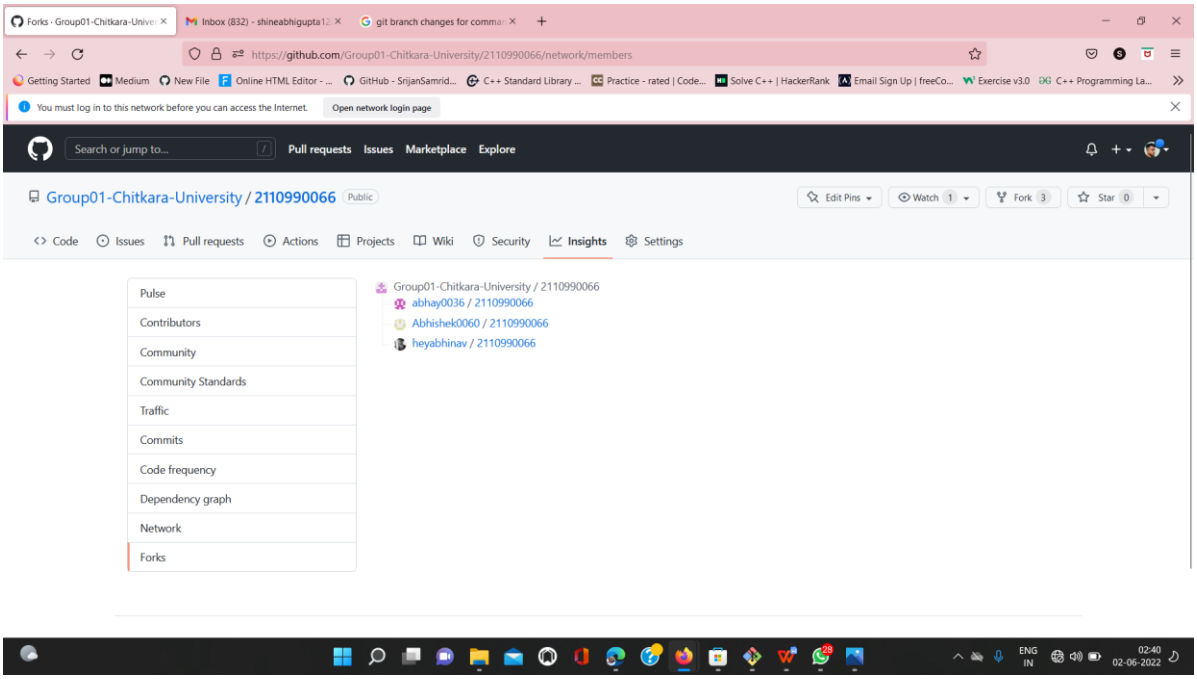
Forks are listed alphabetically by the username of the person who forked the repository

Clicking the number of forks shows you the full network. From there you can click "members" to see who forked the repo

1. On GitHub.com, navigate to the main page of the repository.
2. Under your repository name, click **Insights**.



3. In the left sidebar, click **Forks**.



Here you can see all the forks

Viewing the dependencies of a repository

You can use the dependency graph to explore the code your repository depends on.

Almost all software relies on code developed and maintained by other developers, often known as a supply chain. For example, utilities, libraries, and frameworks. These dependencies are an integral part of your code and any bugs or vulnerabilities in them may affect your code. It's important to review and maintain these dependencies.