

AProjectreport  
on  
**“Task2”**  
with  
**SourceCodeManagement**  
(22CS003)

**Submitted by:**Name:  
Dikshant sohal  
Roll no. 2210990283

**Submitted To:**

Faculty Name: Prabhjot kaur  
chahal  
Department of Computer Science  
&Engineering, Chitkara University  
Institute ofEngineering and  
TechnologyRajpura,Punjab

TeamMember1 Name: Dikshant                      RollNo. 2210990283

TeamMember2Name:Dhruv kinger                      Roll No. 2210990275

TeamMember3Name:Dhruv                      Roll No.2210990277

TeamMember4Name:devesh                      Roll No.2210990265

Institute/School:- **ChitkaraUniversityInstituteofEngineering and  
Name Technology**

DepartmentName: -**DepartmentofComputerScience&Engineering**

ProgramName:- **BachelorofEngineering(B.E.),ComputerScience&Engineering**

CourseName:- **SourceCode  
Management** Session:**2022-23**

CourseCode:- **22CS003** Batch: **2022**  
-

VerticalName:- **First Year** GroupNo:- **G22-A**

**FacultyName:Prof./Dr./Mr./Ms. Dr,prabhjot kaur chahal\_\_\_\_\_**

**Signature:**

**Date:**

## Table of Content

S.NO	Title	PageNo.
1.	Version control with Git	37-39
2.	Problemstatement	40-41
3.	Objective	42
4.	Resources Requirements – FrontEnd/ Backend	43-46
5.	Concepts and commands	47-49
6.	Workflow and Discussion	50
7.	Reference	51

## List of programs in task 1.2

S.No	Program title	Page No	Remark
1	Add collaborators on GitHub repository	25-27	
2	Fork and commit	28-30	
3	Merge and resolve conflicts created due to own activity And collaborator activity	31-32	
4	Reset and revert	33-36	

## **Experiment -1**

### **Aim : Version control with Git**

*What is Git and why it is 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 , creator of the Linux operating system kernel.*

Git is a speedy and efficient distributed VCS tool that can handle project of any size from small to very large ones. Git provide cheap local branching , convenient staging area and multiple workflows. It is free, open source software that lower the cost because developer can use git without paying money. It provide support for non-linear development . Git enable multiple developer or teams to work Separately without having impact on the work of other .

Git is example of of a distributed version control system (DVCS) (hence distributed version control system).



What is GITHUB?

It is the world's biggest open source software developer community platform where the user upload their project using software Git.



What is the difference between GIT and GITHUB?



### What is Repository?

A repository is a directory or storage space where your project can live. Sometimes GitHub user 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 file ,image file ,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 the track of every modification in special kind of database that is accessible to version control software. It helps you to revert back to a 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.

### Type of VCS

1. Local version control system
2. Centralized version control system
3. Distributed version control system

**1. Local Version control System :** Local version control is located in your local machine . if 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 version made after that will be lost.

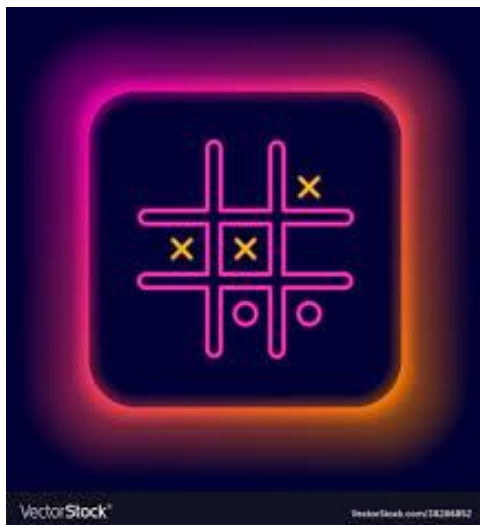
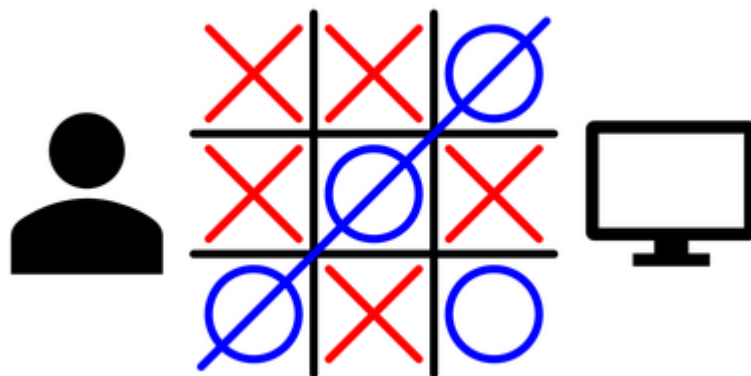
**2. Centralized Version control system :** In the centralized version control system , there will be a single central server that contain all the files related to project and many collaborators checkout files from the single server (you will only have a working copy).The problem is centralized version control system is if the central server crashes , almost everything related to project will be lost.

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

## Experiment – 2

### Aim : Problem Statement

In today's rapidly changing world everyone is running after their goals, such that people forget to spend a leisure time with their family and friends. Many people remain stressed out through there life because of the unwanted work load and peer pressure. We know that life is full of problems but those problems can be felt less stressful if there were some less time consuming and user friendly games which could add a little element of fun within the lives of people during those hard days. Also there was a need of a game which could help in developing skills in children and shape their imagination..





### Experiment – 3

### Solution :

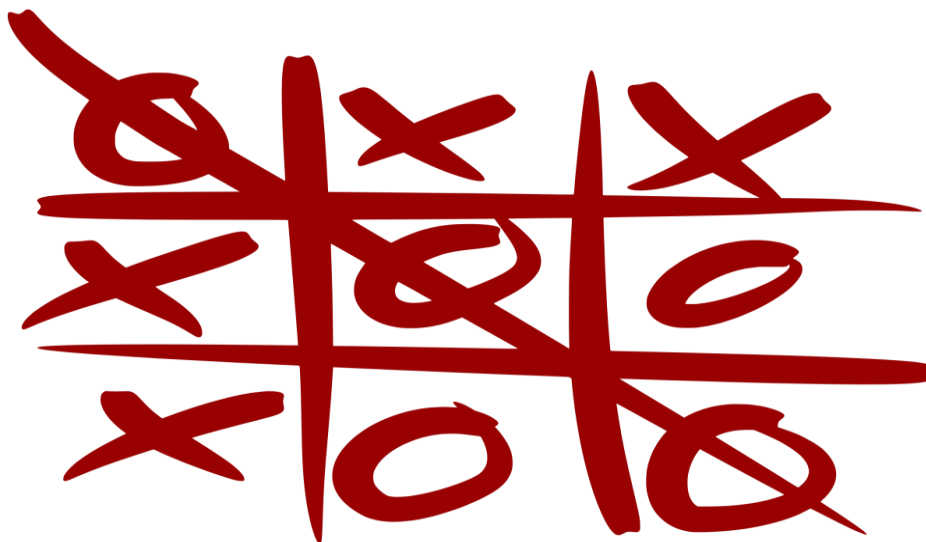
These tic tac toe panels improve hand eye coordination and encourage better social interaction by better collaborative play. And children shouldn't just play with other children. Parents also have a role in the playground more than being supervisors.

The game is played on a grid that's 3 squares by 3 squares.

You are X, your friend (or the computer in this case is zero).

Players take turns putting their marks in empty squares.

The first player to get 3 of her marks in a row (up, down, across, or diagonally) is the winner. When all 9 squares are full, the game is over. You can strive to be on a team, not be the best, not win, but still score a personal victory by not being the worst, allowing you to further skills by being on the team in the first place. You did not win, but you did not lose. The interesting thing in Tic Tac Toe is that the draw occurs only by striving to not lose.



## **Experiment – 4**

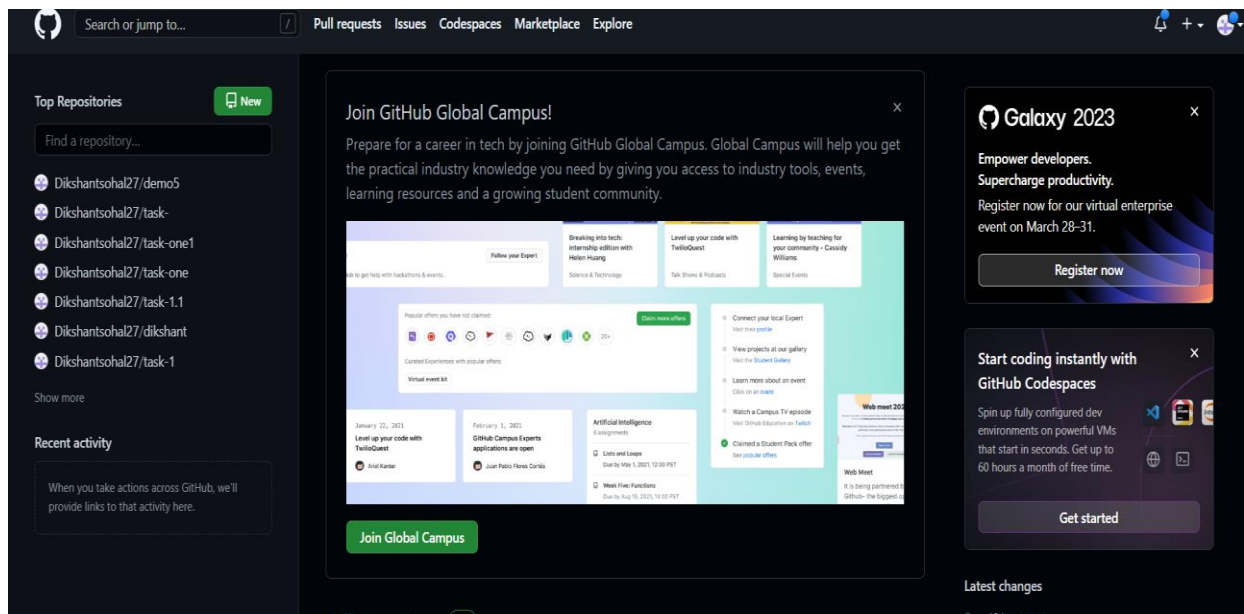
### **Objective**

- 1.Resources Requirement – Frontend/backend**
- 2.Concept and Command**
- 3.Workplace and discussion**
- 4.Reference**

## Experiment – 5

### Aim : Concept and command

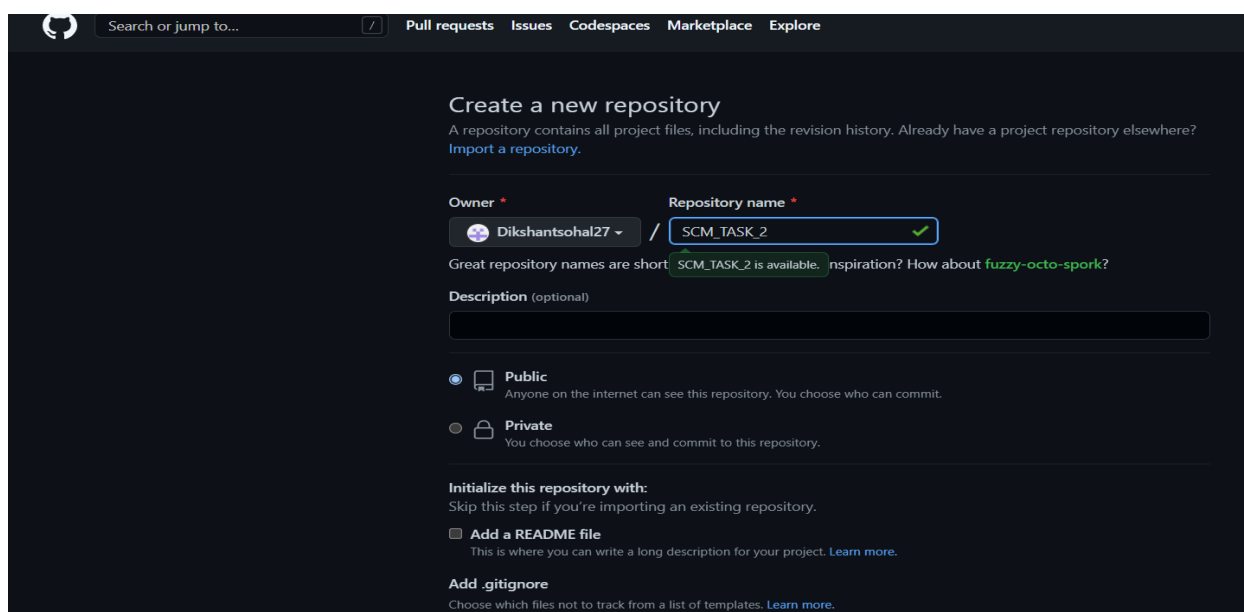
1.Login to your GitHub account and you will land on the homepage as shown below . Click on repositories option in the menu bar.



2.Click on the “New” button in the top right corner.

3.Enter the repository name and add the description of the repository.

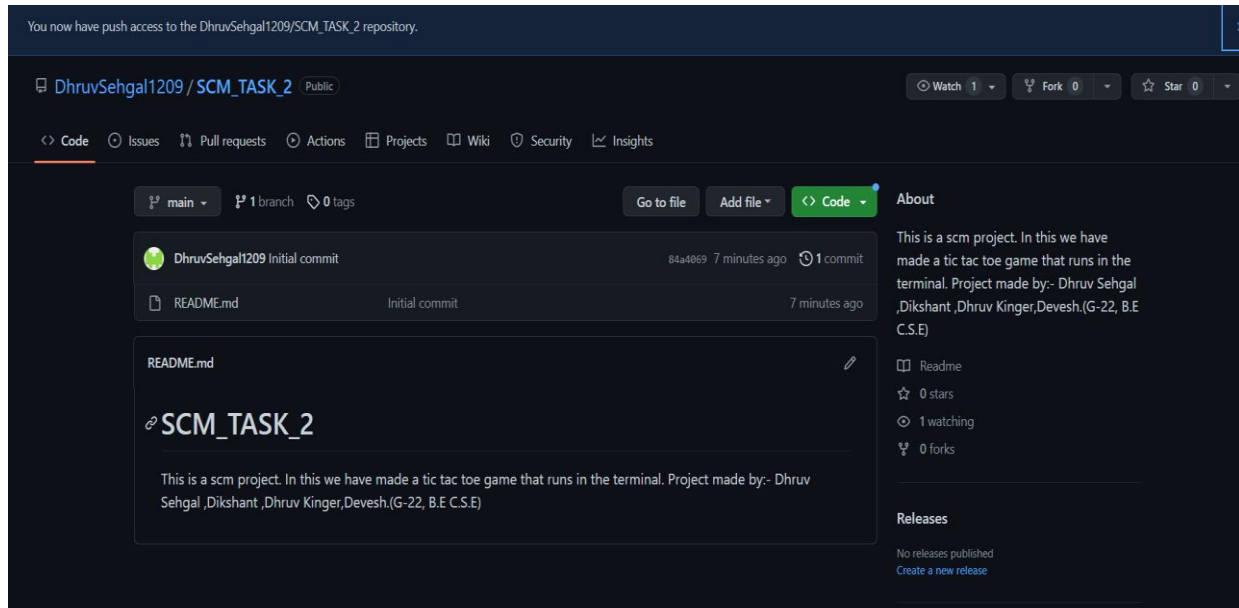
4.Select if you want the repository to public or private.



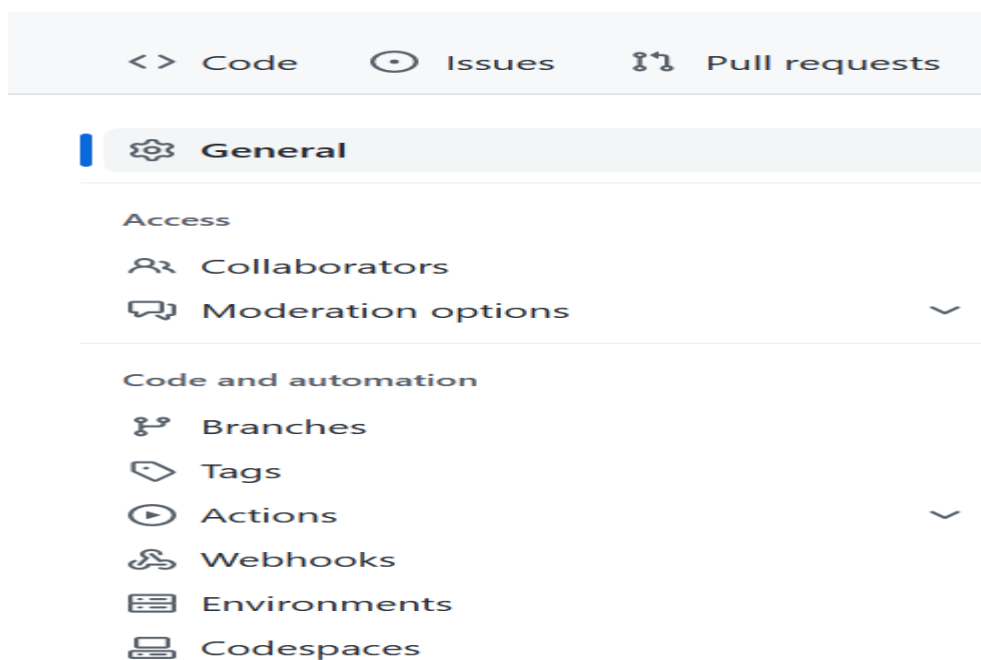
5.If you want to import code from the existing repository select the import code option.

6.Now, you have created your repository successfully.

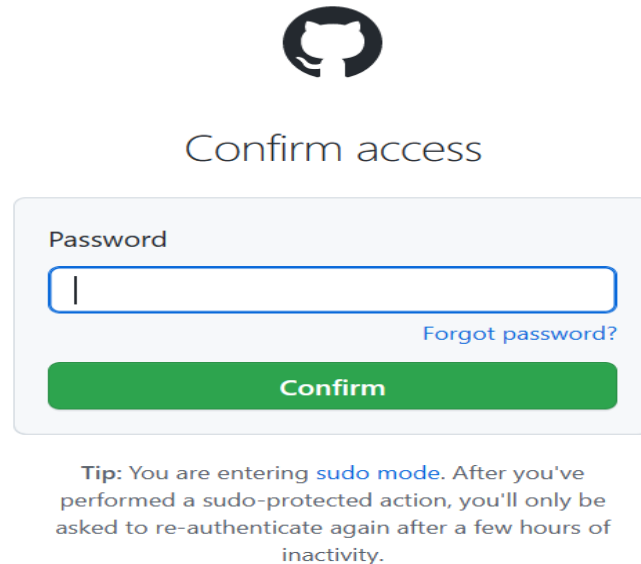
7.To add member to your repository open the repository and select setting option in the navigation bar.



8.Click on collaboration option under the access tab.



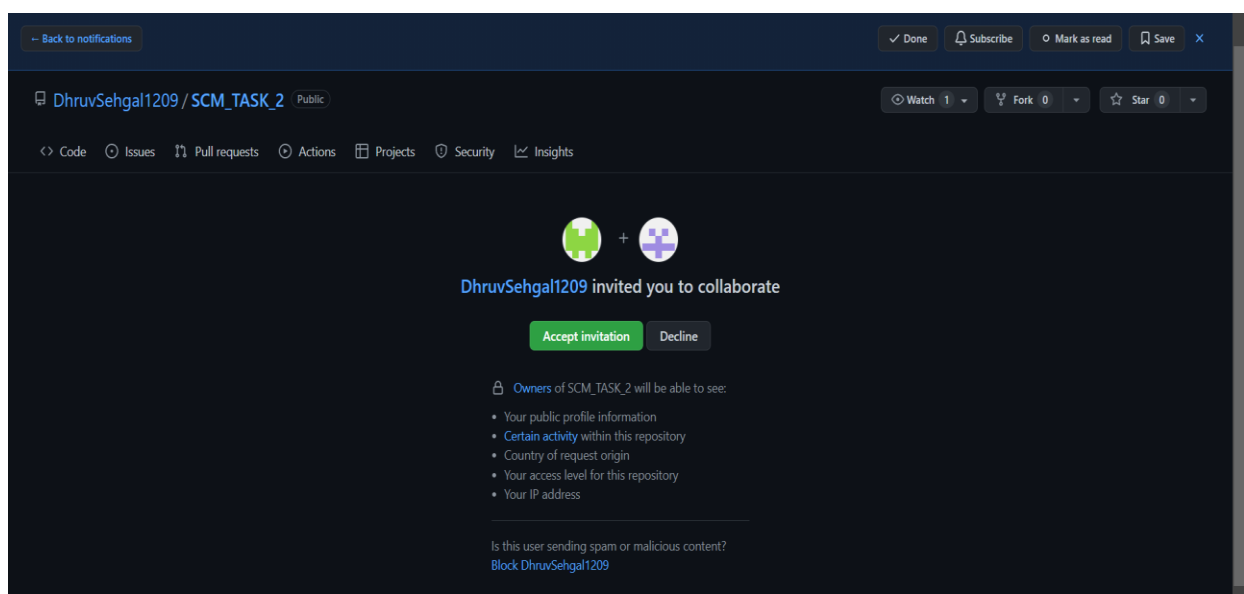
9. After clicking on collaboration GitHub ask you to enter your password to confirm access to the repository.



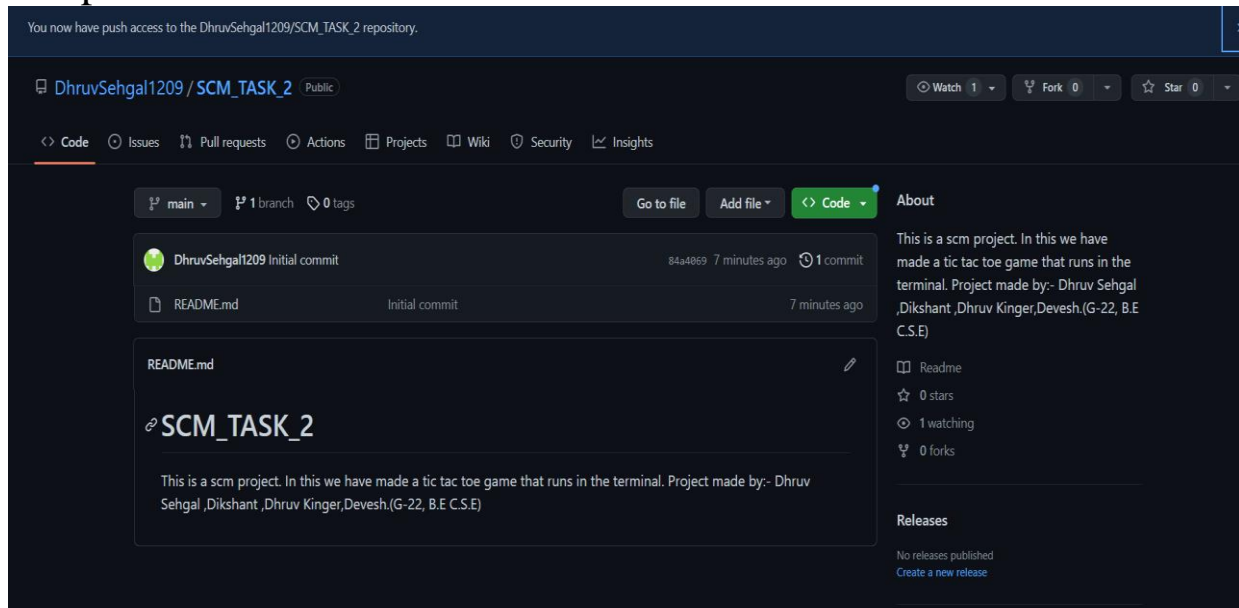
The image shows the GitHub 'Confirm access' dialog. At the top is the GitHub logo. Below it, the text 'Confirm access' is centered. A light gray box contains a 'Password' label, a text input field with a cursor, and a 'Forgot password?' link. Below the input field is a green 'Confirm' button. At the bottom, a tip states: 'Tip: You are entering sudo mode. After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity.'

10. After entering the password you can manage access and add/remove team member to your project.

11. To add member click on the add people and search the id of your respective team member.



15. You will be redirected to GitHub where you can either select to accept or decline the invitation.



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

17. Now all member are ready to contribute to the project.

18. To open a pull request we first have to make a new branch by using git branch branchname option.

```
diksh@hp MINGW64 ~/Desktop/New folder (checkout_page)
$ ssh-keygen -t ed25519 -C "dikshant283.be22@chitkara.edu.in"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c:/Users/diksh/.ssh/id_ed25519):
Created directory '/c:/Users/diksh/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c:/Users/diksh/.ssh/id_ed25519
Your public key has been saved in /c:/Users/diksh/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:A4vIye4ESicNkLv8UDYY/r60QGIXniffZd7RSrnUbuw dikshant283.be22@chitkara.edu.in
The key's randomart image is:
+--[ED25519 256]--+
|..      .. 0..|
|0 .      .. 0+..|
|0 . = = .. 0..|
|..000 *..+0 + .. 0..|
|0+0+0+0.5   0 |
|++0.0 ..   E|
|.. + ....   |
|0 . 0..    |
| . 0..     |
+----[SHA256]-----+

diksh@hp MINGW64 ~/Desktop/New folder (checkout_page)
$ eval "$(ssh-agent -s)"
Agent pid 1219

diksh@hp MINGW64 ~/Desktop/New folder (checkout_page)
$ ssh-add ~/.ssh/id_ed25519
bash: ssh-add: ~/.ssh/id_ed25519: No such file or directory

diksh@hp MINGW64 ~/Desktop/New folder (checkout_page)
$ ssh-add ~/.ssh/id_ed25519
bash: ssh-add: ~/.ssh/id_ed25519: No such file or directory

diksh@hp MINGW64 ~/Desktop/New folder (checkout_page)
$ A[[200-ssh-add ~/.ssh/id_ed25519
bash: $'[[200-ssh-add': command not found

diksh@hp MINGW64 ~/Desktop/New folder (checkout_page)
$ ssh-add ~/.ssh/id_ed25519
Identity added: /c:/Users/diksh/.ssh/id_ed25519 (dikshant283.be22@chitkara.edu.in)

diksh@hp MINGW64 ~/Desktop/New folder (checkout_page)
$ clip < ~/.ssh/id_ed25519.pub
```

19. After making new branch we add a file to the branch or make changes in the existing file.

```
diksh@hp MINGW64 ~/Desktop/New folder (checkouot_page)
$ git clone git@github.com:Dikshantsohal127/TASK-2.git
Cloning into 'TASK-2'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 15 (delta 2), reused 4 (delta 1), pack-reused 0
Receiving objects: 100% (15/15), 4.06 KiB | 2.03 MiB/s, done.
Resolving deltas: 100% (2/2), done.

diksh@hp MINGW64 ~/Desktop/New folder (checkouot_page)
$ ls
TASK-2/

diksh@hp MINGW64 ~/Desktop/New folder (checkouot_page)
$ cd TASK-2/

diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (main)
$ touch scm1.txt

diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (main)
$ git add scm1.txt

diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   scm1.txt

diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (main)
$ git commit -m "new file added"
[main 996cbf0] new file added
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 scm1.txt
```

20. Add and commit the changes to the local repository.

```
MINGW64/c/Users/diksh/Desktop/New folder/TASK-2
TASK-2/

diksh@hp MINGW64 ~/Desktop/New folder (checkouot_page)
$ cd TASK-2/

diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (main)
$ touch scm1.txt

diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (main)
$ git add scm1.txt

diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   scm1.txt

diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (main)
$ git commit -m "new file added"
[main 996cbf0] new file added
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 scm1.txt

diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (main)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 237 bytes | 237.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Dikshantsohal127/TASK-2.git
   b74f758..996cbf0 main -> main
```

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

```
MINGW64~/Users/diksh/Desktop/New folder/TASK-2
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 237 bytes | 237.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Dikshantsohal127/TASK-2.git
b74f758..996cbf0 main -> main

diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (main)
$ git checkout -b branch_1
Switched to a new branch 'branch_1'

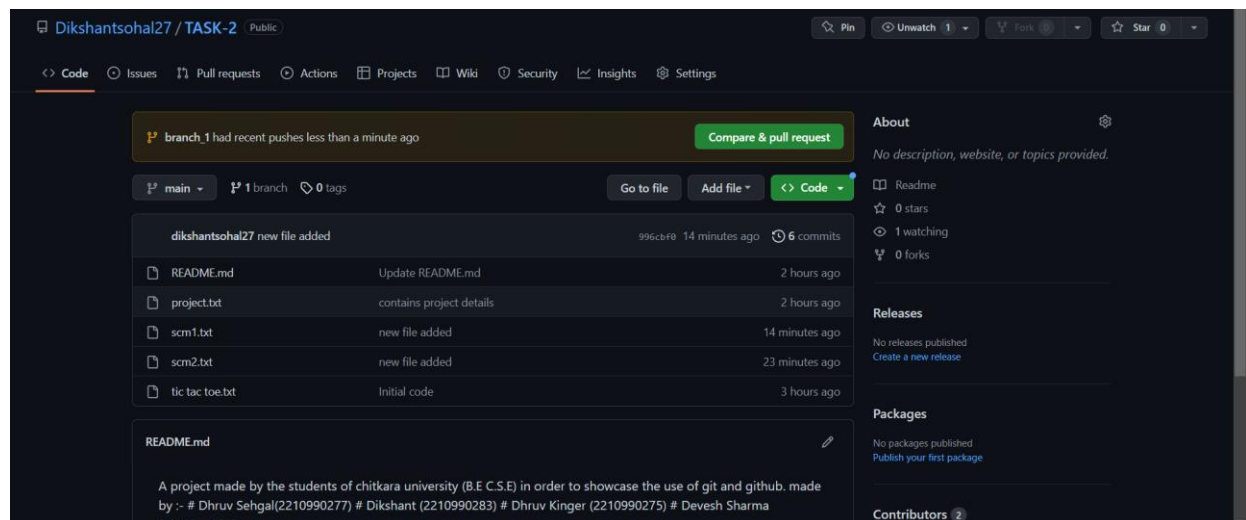
diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (branch_1)
$ git status
On branch branch_1
nothing to commit, working tree clean

diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (branch_1)
$ cat > scm2.txt
this is my scm project.
hello
bye
tc

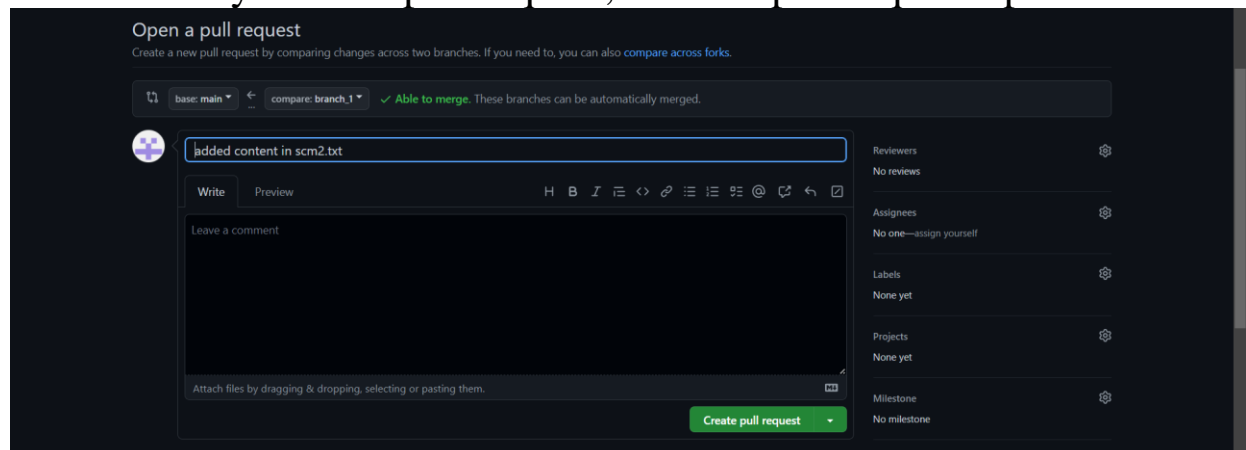
diksh@hp MINGW64 ~/Desktop/New folder/TASK-2 (branch_1)
$ git status
On branch branch_1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   scm2.txt

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

22. After pushing new branch GitHub will either automatically ask you to create a pull request or you can create your own pull request

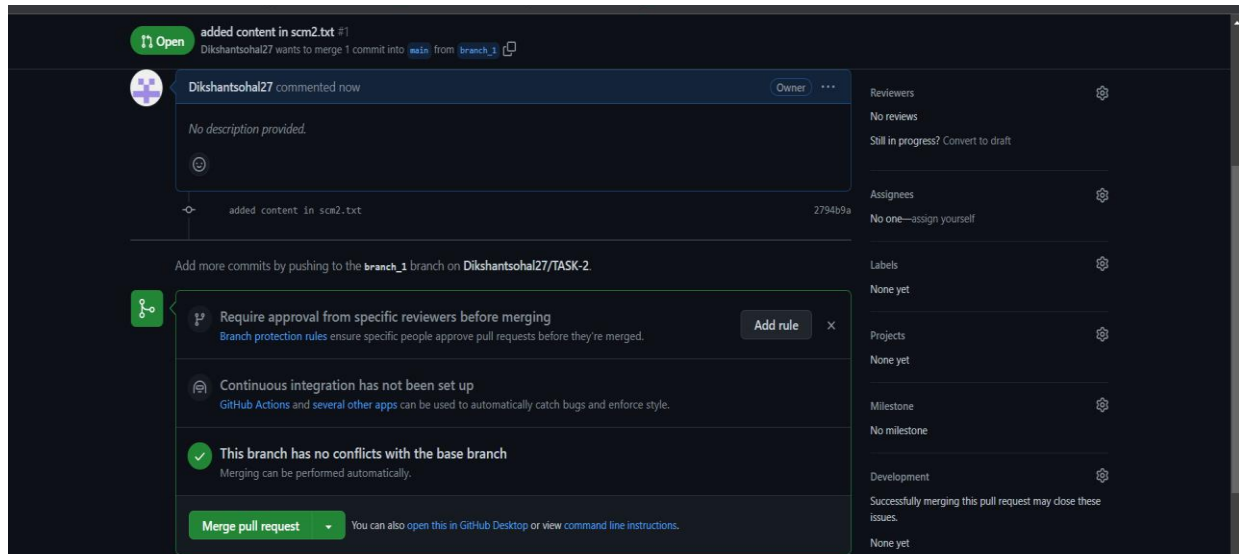


23. To create your own pull request, click on pull request option.





**24.**GitHub will detect any conflict and ask you to enter a description of your pull request .

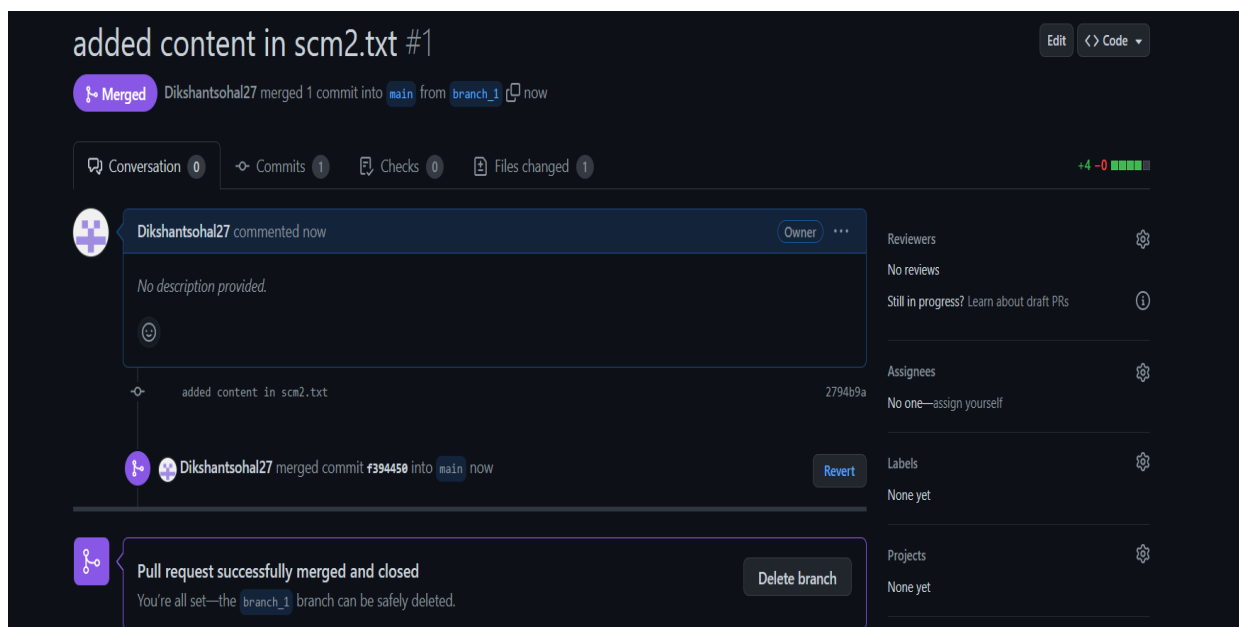


**25.**After opening a pull request all the members will be sent the request if they want to merge or close the request.

**25.**If the team member choses not to merge your pull request they will close the pull request.

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

**27.**You can see all the pull request generated and how they were deal with clicking the pull request.



## **Experiment : 6**

### **Workplace and discussion :**

1)Project Overview: The project is about the game “TIK TAC TOE” which is a very easy to play and useful for sharpness of the mind.

2)Technical details : In this project we make the backend of the game tik tac toe by using the python language.

3)Feature and functionality : In this project we use the “Graphical user interface” which helps to enhance the project and make the project more Creative and interesting for the user to play.

4)Conclusion : It is very easy to play this game , secondly there is no wastage of paper as you are playing this game in your laptop or computer.

### **Experiment :7**

### ***References :***

(<https://devdojo.com/jothin-kumar/tic-tac-toe-with-python-tkinter-part-1>)we have used this code for the backend of the project.