

A Project report on

**“LEAP – Leisure, Entertainment and Pleasure”**

with

**Source Code Management**

**(CS181)**

Submitted by

<b>Team Member 1</b>	<b>Dhruv Arora</b>	<b>2110992073</b>
<b>Team Member 2</b>	<b>Prince Gupta</b>	<b>2110992024</b>
<b>Team Member 3</b>	<b>Priyanshu Garg</b>	<b>2110992065</b>
<b>Team Member 4</b>	<b>Kalpana</b>	<b>2110992096</b>



**Department of Computer Science & Engineering**  
Chitkara University Institute of Engineering and Technology, Punjab

Jan- June  
(2021-22)

Institute/School Name	<b>Chitkara University Institute of Engineering and Technology</b>		
Department Name	<b>Department of Computer Science &amp; Engineering</b>		
Programme Name	<b>Bachelor of Engineering (B.E.), Computer Science &amp; Engineering</b>		
Course Name	<b>Source Code Management</b>	Session	<b>2021-22</b>
Course Code	<b>CS181</b>	Semester/Batch	<b>2<sup>nd</sup>/2021</b>
Vertical Name	<b>Zeta</b>	Group No	<b>G0X</b>
Course Coordinator	<b>Dr. Neeraj Singla</b>		
Faculty Name	<b>Dr Anuj Jain</b>		

## Submission

Name:

Signature:

Date:

## Table of Content

<b>S. No.</b>	<b>Title</b>	<b>Page No.</b>
1	Version control with Git	3-5
2	Problem Statement	6
3	Objective	6
4	Resources Requirements – Frontend / Backend	7
5	Concepts and commands, Workflow	8-20

## 1. Version control with Git

### 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?

<b>GIT      VERSUS      GITHUB</b>	
Git is a distributed version control system which tracks changes to source code over time.	GitHub is a web-based hosting service for Git repository to bring teams together.
Git is a command-line tool that requires an interface to interact with the world.	GitHub is a graphical interface and a development platform created for millions of developers.
It creates a local repository to track changes locally rather than store them on a centralized server.	It is open-source which means code is stored in a centralized server and is accessible to everybody.
It stores and catalogs changes in code in a repository.	It provides a platform as a collaborative effort to bring teams together.
Git can work without GitHub as other web-based Git repositories are also available.	GitHub is the most popular Git server but there are other alternatives available such as GitLab and BitBucket.
	

## 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.
  
- 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.

## 2. Problem Statement

Listening to music, podcasts, reading e-books, or getting the right news feed on the web can sometimes become difficult and inaccessible owing to repetitive and distracting adverts, and occasionally due to the paid policy of these programs. These websites' user interfaces and user experiences are both poor (UX). Apart from this you definitely can't do all this in a single website, you'll need to open and register on multiple websites to access these.

## 3. Objective

PROJECT NAME: LEAP – LEISURE, ENTERTAINMENT AND PLEASURE

Our goal is to create a web app for our users that will provide them with a fresh experience with music, podcasts, news, e-books, and other content. Our prime objective is to create a lightweight online app with a very basic yet classic UI/UX that allows our customers to listen to music, podcasts, or read e-books and news on the same platform without paying a large fee or being bombarded with annoying adverts.

## 4. Resources Required.

Frontend – HTML5, CSS3, Javascript

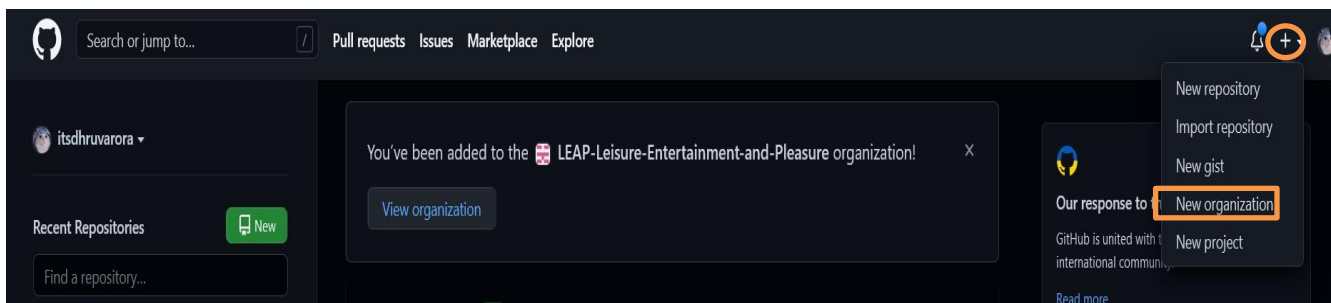
Backend – NodeJS



## 5. Concepts and commands, Workflow

**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 the button shown in the menu bar and then click on New Organization.



- Set Up Your Organization. Fill Your Organization's Name and Other Details.

Tell us about your organization

## Set up your organization

**Organization account name \***

LEAP - Leisure, Entertainment and Pleasure ✓

This will be the name of your account on GitHub.  
Your URL will be: <https://github.com/LEAP-Leisure-Entertainment-and-Pleasure>.

**Contact email \***

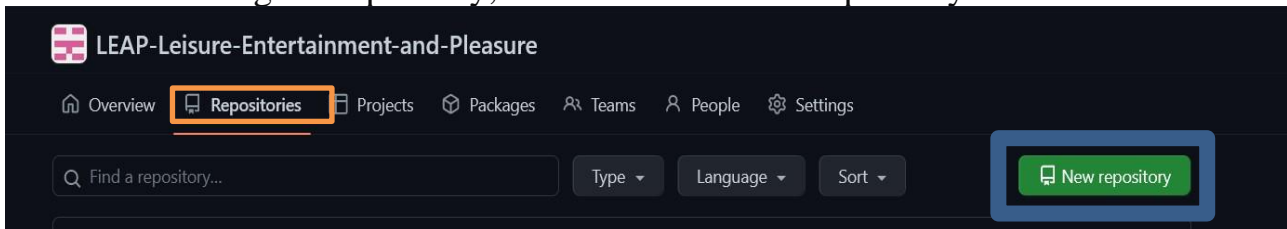
**This organization belongs to: \***

☒ **My personal account**  
I.e., itsdhravarora (Dhruv Arora)

☐ **A business or institution**  
For example: GitHub, Inc., Example Institute, American Red Cross



➤ After creating the repository, we have to create a Repository.



➤ Create a New Repo by Pressing the New repository button. Fill in the required details.

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

**Owner \*** LEAP-Leisure-Entertainment-and-Pleasure / **Repository name \*** LEAP-main ✓

Great repository names are short and memorable. Need inspiration? How about [friendly-adventure?](#)

**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**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

License: GNU General Public Li...

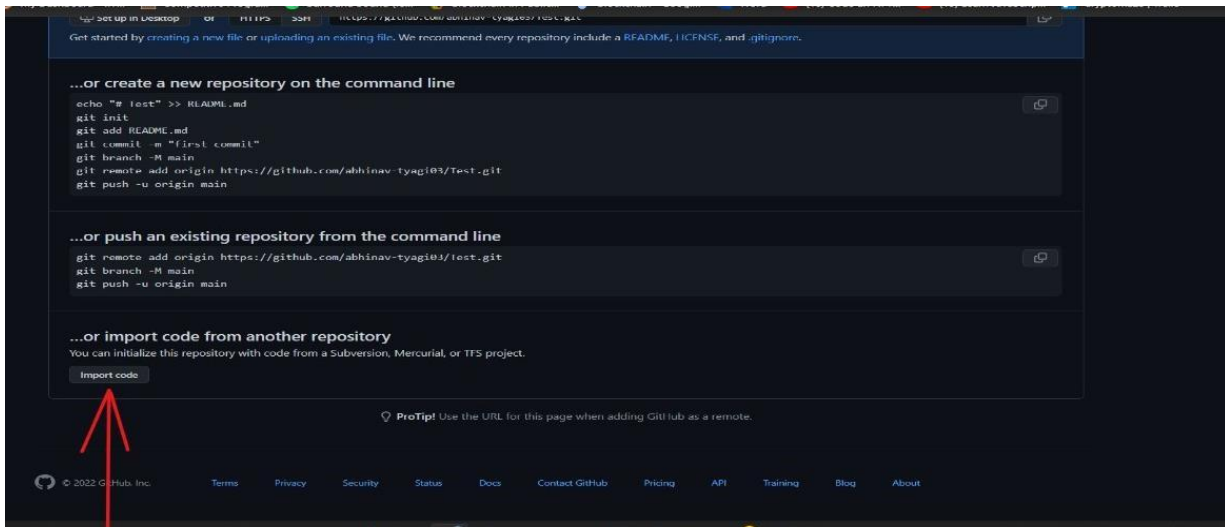
This will set main as the default branch. Change the default name in LEAP-Leisure-Entertainment-and-Pleasure's [settings](#).

---

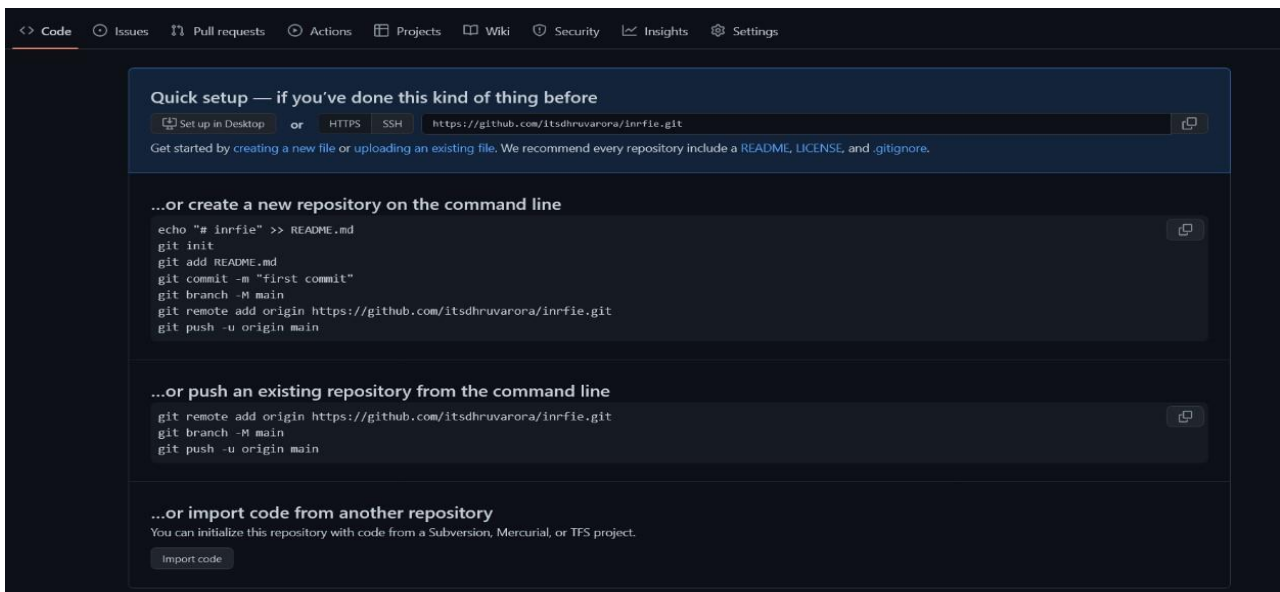
📘 You are creating a public repository in the LEAP-Leisure-Entertainment-and-Pleasure organization.

**Create repository**

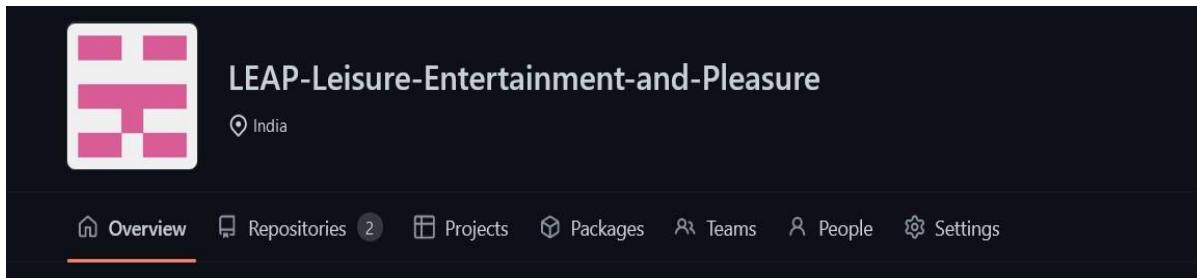
- If you want to import code from an existing repository select the import code option.



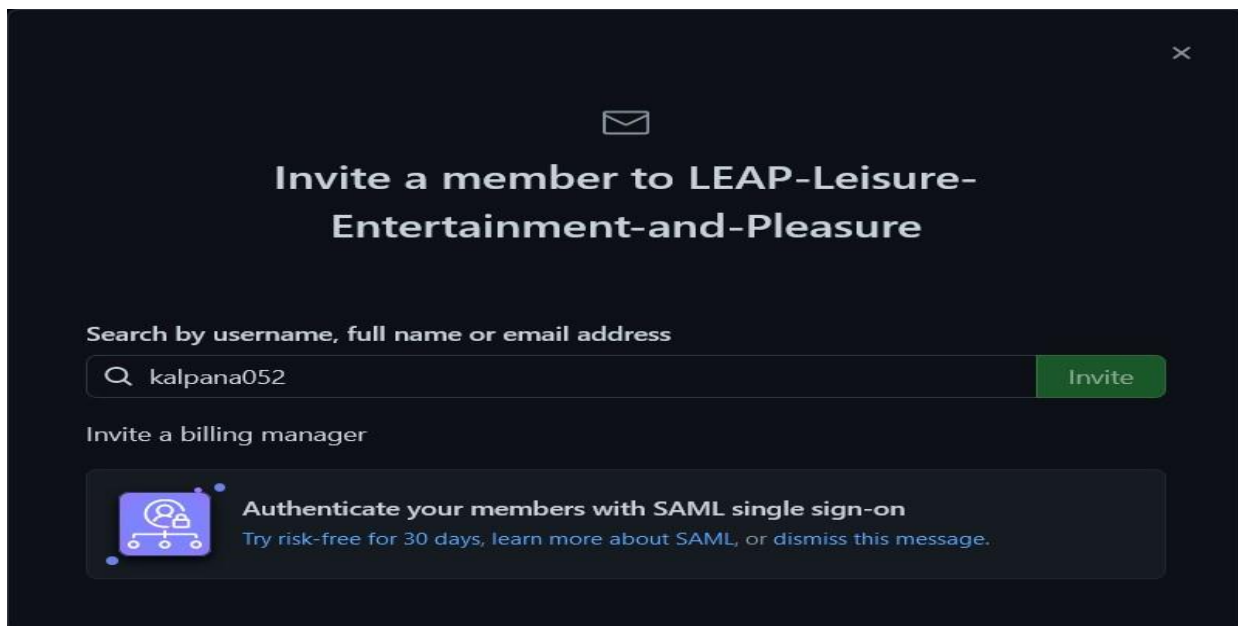
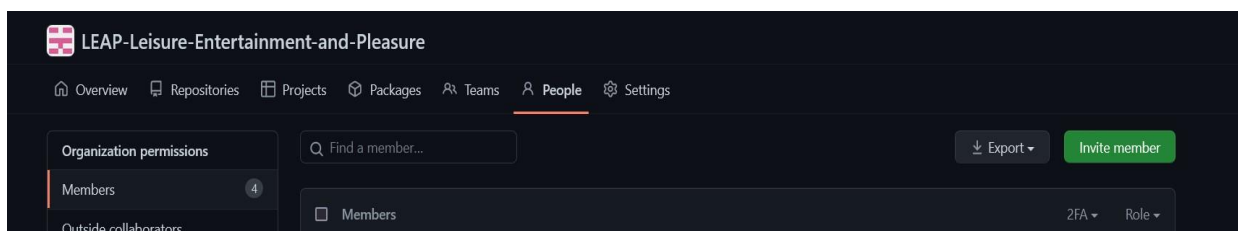
- To create a new file or upload an existing file into your repository select the option in the following box.



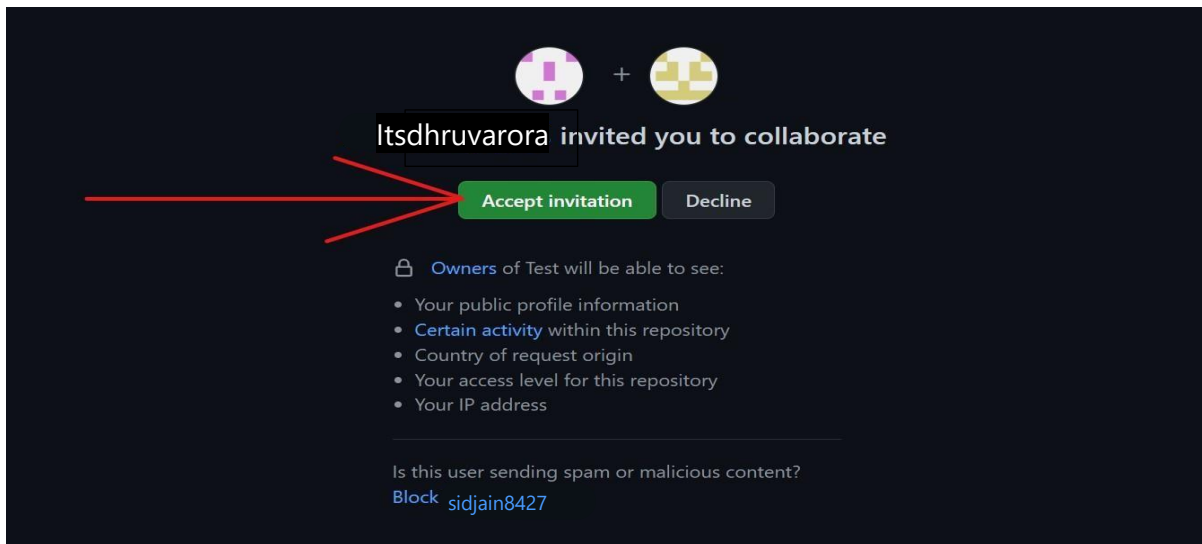
- Now, you have created your repository successfully.
- To add members to your repository, open your Organization and select People option in the navigation bar.
- Click on Collaborators option under the access tab.



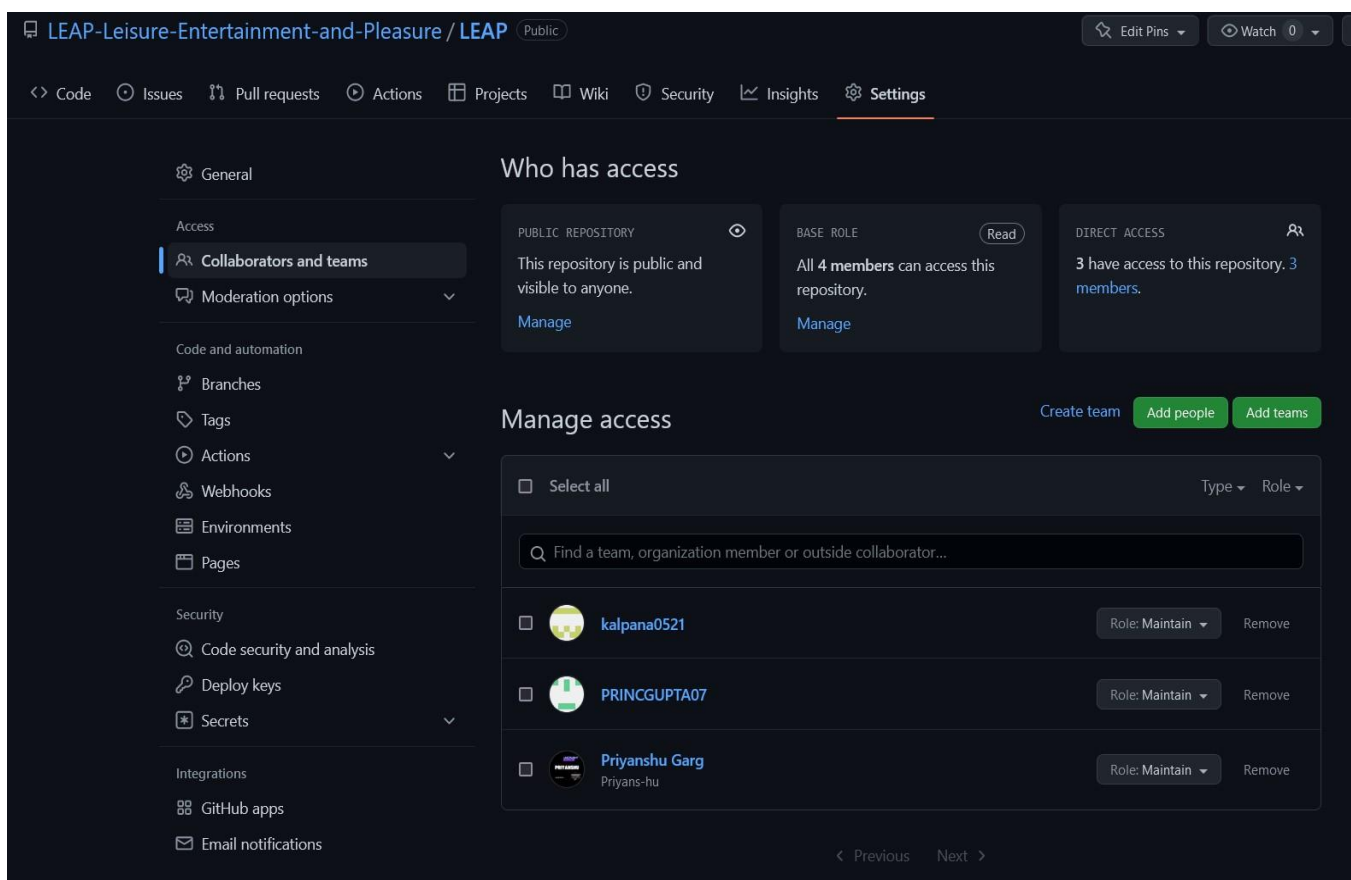
- To add members click on the add people option and search the id of your respective team member.



- 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.



- Next, Open the desired Repository in the Organisation. Look and click on Settings -> Collaborators and Teams. Here you can Manage the role of each collaborator.



## Experiment No. 02

- To Open a Pull Request, First of All, it will be required to fork the repository and commit changes into your own.

```
dhruv@LAPTOP-QPLL060F MINGW64 /e
$ cd D:

dhruv@LAPTOP-QPLL060F MINGW64 /d
$ mkdir LEAP-PROJECT

dhruv@LAPTOP-QPLL060F MINGW64 /d
$ cd LEAP-PROJECT

dhruv@LAPTOP-QPLL060F MINGW64 /d/LEAP-PROJECT
$ git init
Initialized empty Git repository in D:/LEAP-PROJECT/.git/

dhruv@LAPTOP-QPLL060F MINGW64 /d/LEAP-PROJECT (master)
$ git clone https://github.com/itsdhruvarora/LEAP.git
Cloning into 'LEAP'...
remote: Enumerating objects: 174, done.
remote: Counting objects: 100% (93/93), done.
remote: Compressing objects: 100% (80/80), done.
remote: Total 174 (delta 13), reused 89 (delta 11), pack-reused 81
Receiving objects: 100% (174/174), 29.75 MiB | 3.06 MiB/s, done.
Resolving deltas: 100% (36/36), done.
```

- Add and commit the changes to the local repository.

```
dhruv@LAPTOP-QPLL060F MINGW64 /d/LEAP-PROJECT (master)
$ git add .
warning: adding embedded git repository: LEAP
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:   git submodule add <url> LEAP
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:   git rm --cached LEAP
hint: See "git help submodule" for more information.

dhruv@LAPTOP-QPLL060F MINGW64 /d/LEAP-PROJECT (master)
$ ls
LEAP/

dhruv@LAPTOP-QPLL060F MINGW64 /d/LEAP-PROJECT (master)
$ cd LEAP

dhruv@LAPTOP-QPLL060F MINGW64 /d/LEAP-PROJECT/LEAP (main)
$ ls
Ebook/  Music/  home/  index.html  index2.html  leap.png  podcast/  readme.md

dhruv@LAPTOP-QPLL060F MINGW64 /d/LEAP-PROJECT/LEAP (main)
$ vi index.html

dhruv@LAPTOP-QPLL060F MINGW64 /d/LEAP-PROJECT/LEAP (main)
$ git add index.html
```

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

```
dhruv@LAPTOP-QPLL060F MINGW64 /d/LEAP-PROJECT/LEAP (main)
$ git add index.html

dhruv@LAPTOP-QPLL060F MINGW64 /d/LEAP-PROJECT/LEAP (main)
$ git commit -m "changed title"
[main 7986a25] changed title
1 file changed, 1 insertion(+), 1 deletion(-)

dhruv@LAPTOP-QPLL060F MINGW64 /d/LEAP-PROJECT/LEAP (main)
$ git remote add leap https://github.com/itsdhruvarora/LEAP.git

dhruv@LAPTOP-QPLL060F MINGW64 /d/LEAP-PROJECT/LEAP (main)
$ git push leap
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 296 bytes | 296.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/itsdhruvarora/LEAP.git
87607e8..7986a25 main -> main
```

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

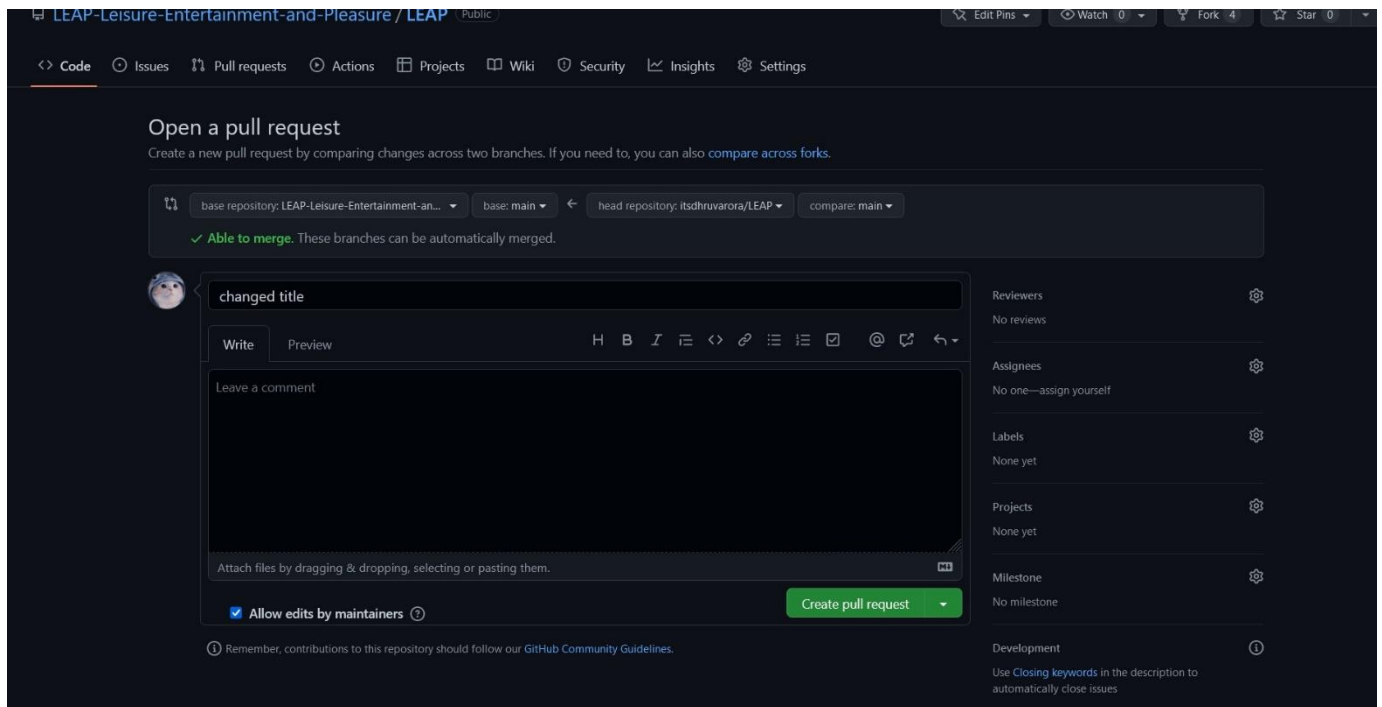
The top screenshot shows the GitHub repository page for 'itsdhruvarora / LEAP'. The repository is public and forked from 'LEAP-Leisure-Entertainment-and-Pleasure/LEAP'. The 'main' branch is selected, and it is up to date with the upstream. The page shows options to view code, pull requests, actions, projects, wiki, security, insights, and settings. There is a 'Code' button and a 'Fetch upstream' button.

The bottom screenshot shows the same repository page, but with a pull request modal open. The modal indicates that the current branch is 1 commit ahead of the upstream 'main' branch. It shows a commit by 'itsdhruvarora' titled 'changed title' and offers the option to 'Open pull request'.

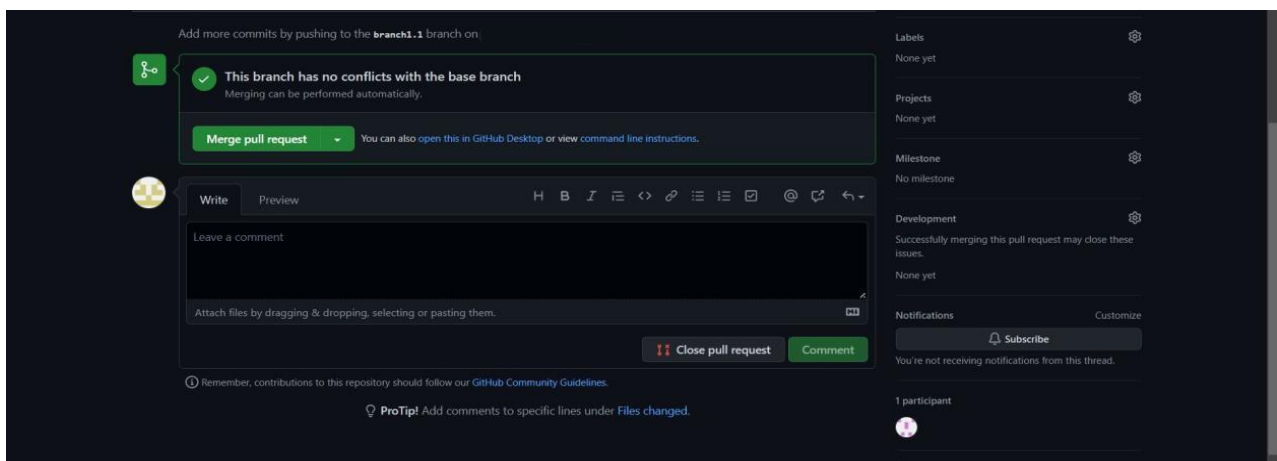
- To create your own pull request, click on pull request option.
- GitHub will detect any conflicts and ask you to enter a description of your pull request.

\

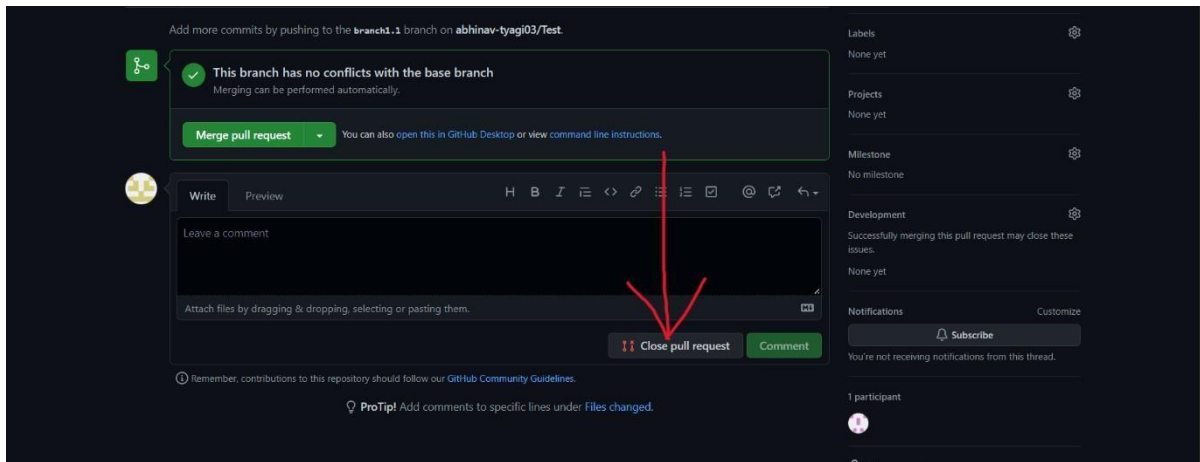




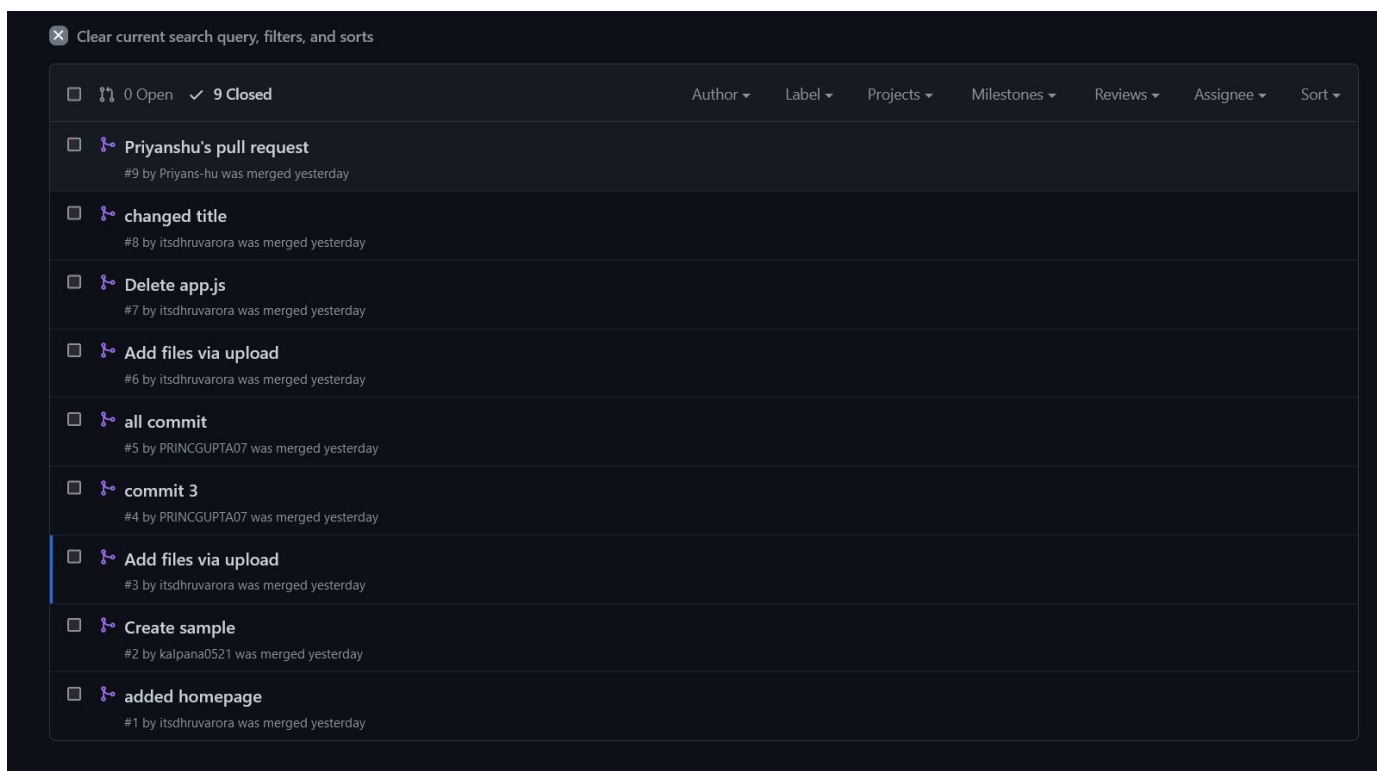
- After opening a pull request all the team members will be sent the request if they want to merge or close the request.



- 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: 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.

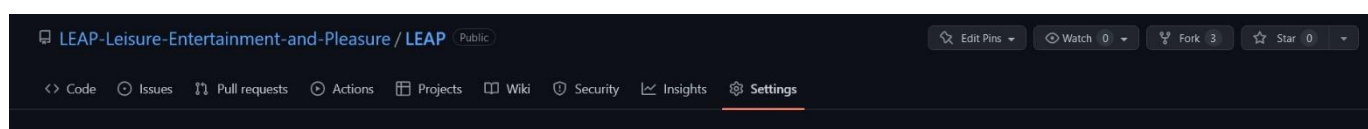
Some repository graphs are available only in public repositories with GitHub Free:

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

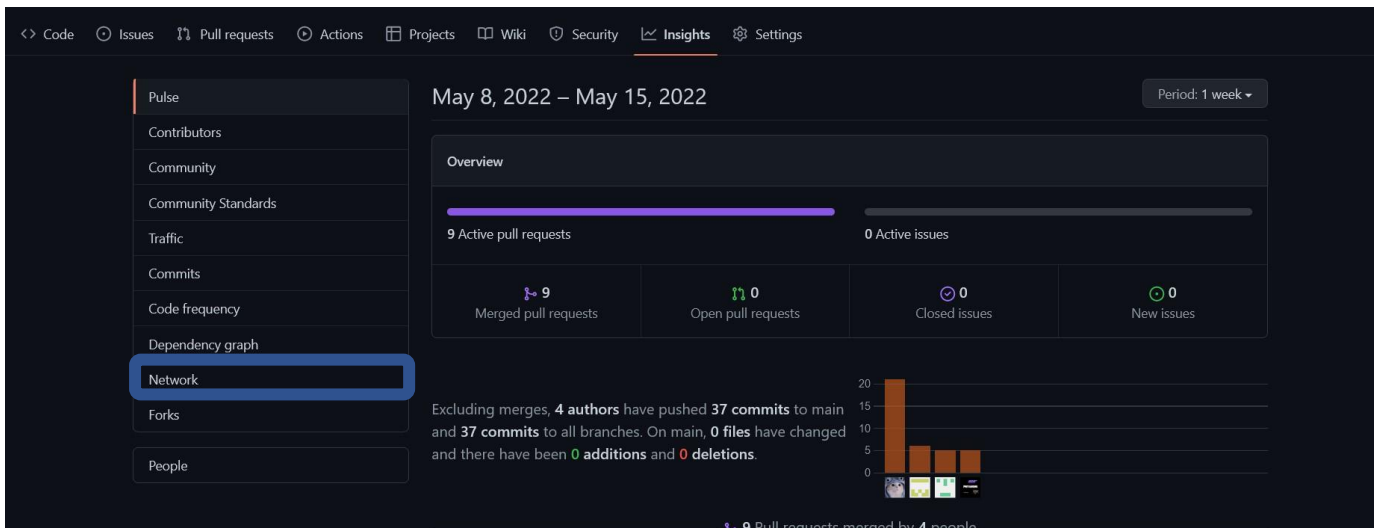
### Steps to access network graphs of respective repository

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.

## Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.

