

A Project report on

"FOOD-52"

with

Source Code Management

(CS181)

Submitted by

Team Member 1	Loveleen Goyal	2110992021
Team Member 2	Sakshi	2110992049
Team Member 3	Kavya Nagpal	2110992030
Team Member 4	Deepankar	2110992046



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	zeta	Group No	G-27
Course Coordinator	Dr. Sachendra Singh		
Faculty Name	Dr. Sachendra Singh		

Submission

Name: Loveleen Goyal

Signature:

Date:



Table of Content

S. No.	Title	Page No.
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	8-22

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?

GIT VS GITHUB	
GIT	GITHUB
Git is a distributed version control system which track 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 which requires an interface to interact with the world.	Github is a graphical interface and a development platform created for millions of developers.
It creates local repository to track changes locally rather than store them on a centralized server.	It is open source which means code is stored on a centralized server.
It stores and catalog changes in code in a repository.	It provides a platform as a collaborative effort to bring teams together.

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

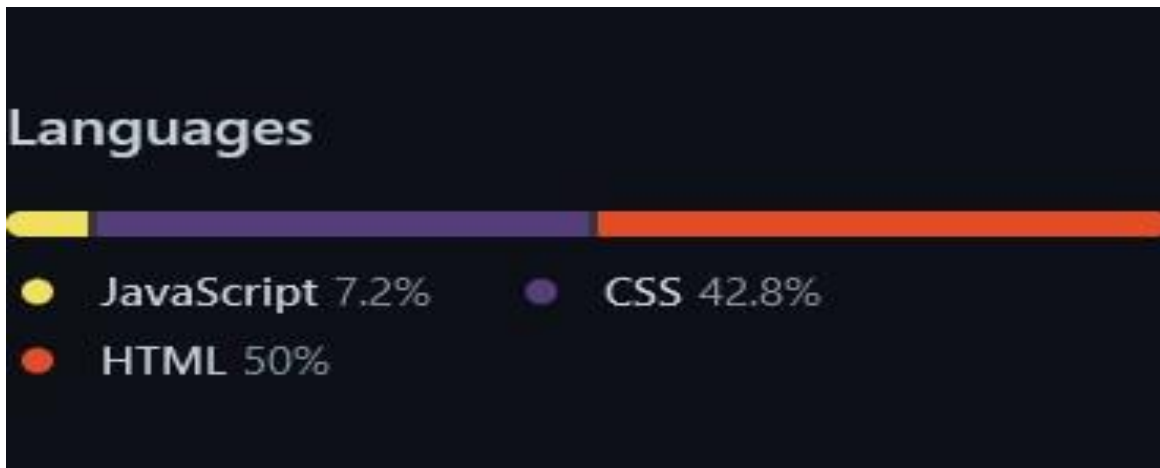
PROJECT NAME: LEAP – LEISURE, ENTERTAINMENT AND PLEASURE

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

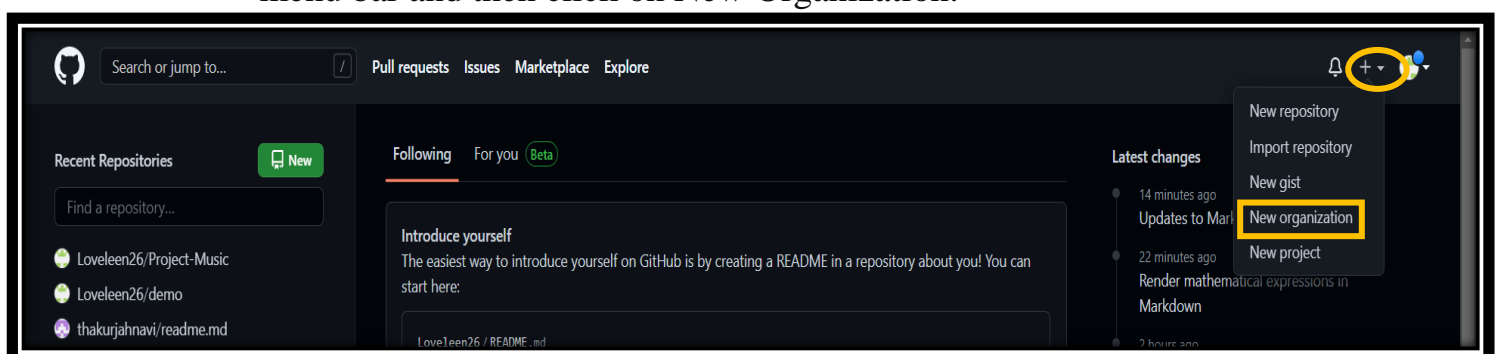
Backend – NodeJS



5. Concepts , Commands, Workflow and Discussions.

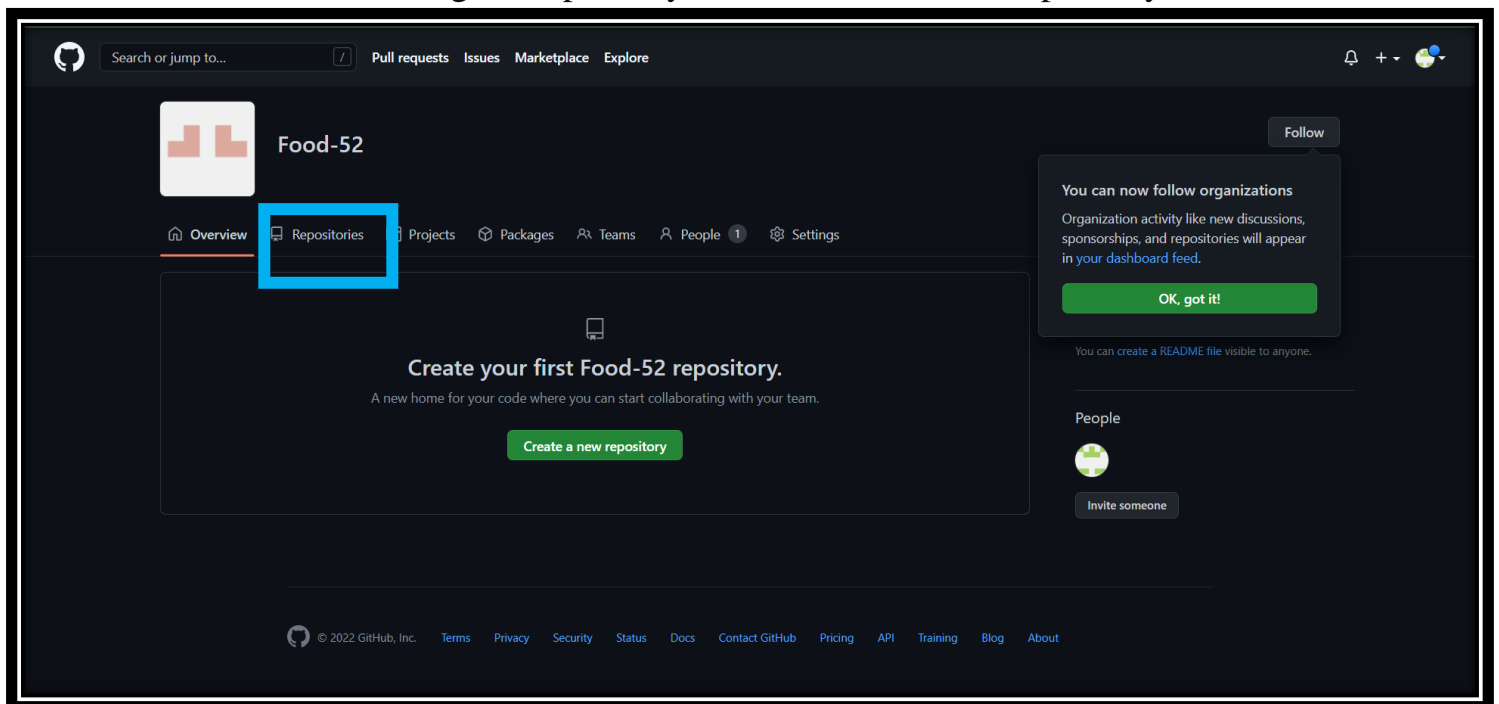
Aim: Create a distributed Repository and add members in project team

- 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. .

- After creating the repository, we have to create a 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 *

Repository name *

Food-52

/

Project_Food-52

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

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: None ▾

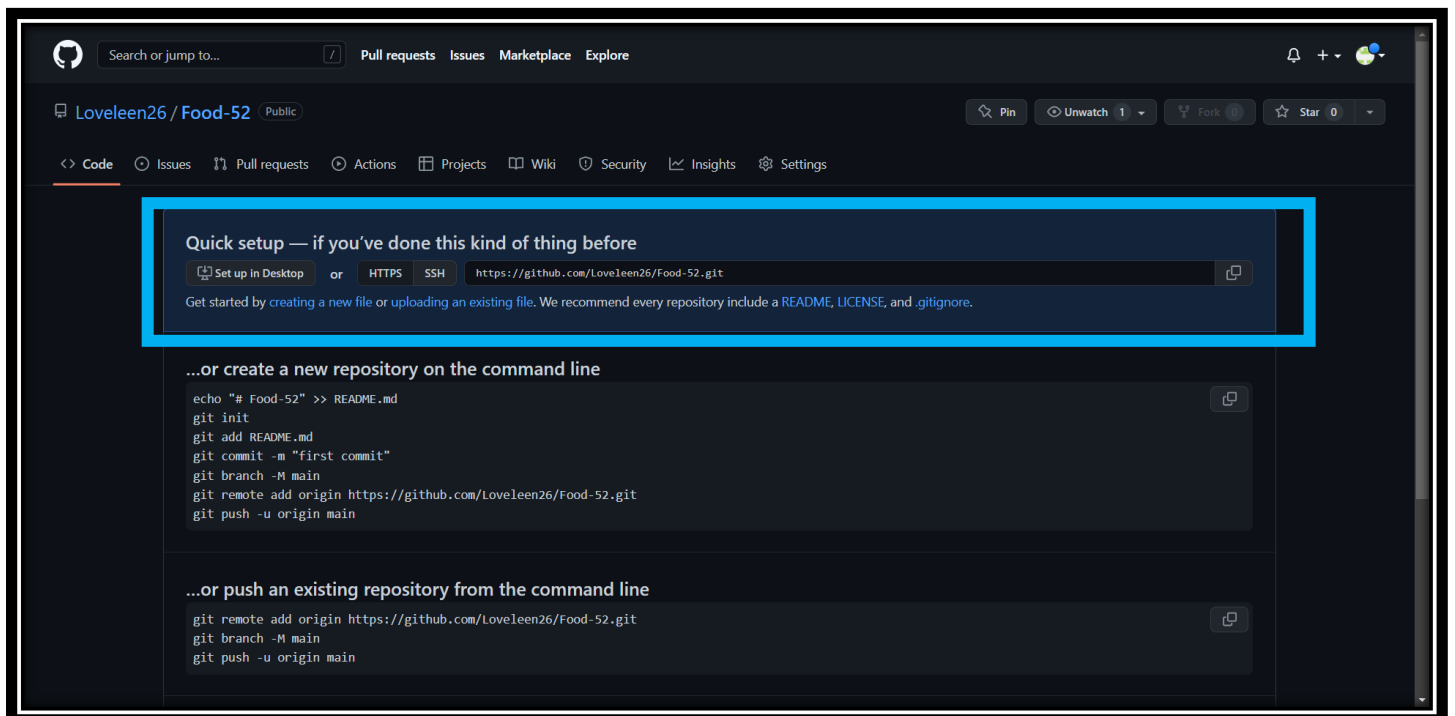
You are creating a public repository in the Food-52 organization.

Create repository

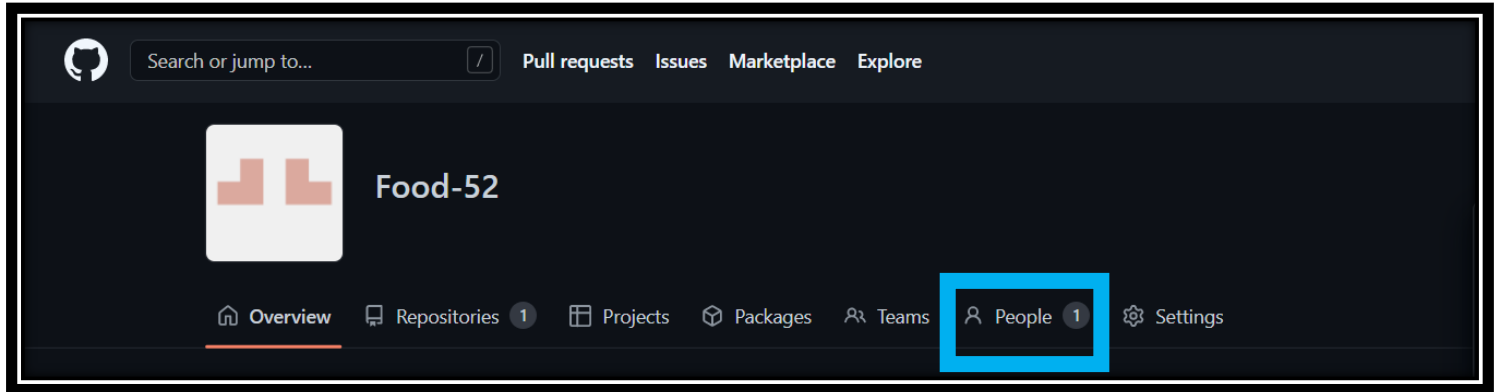
- If you want to import code from an existing repository select the import code option.



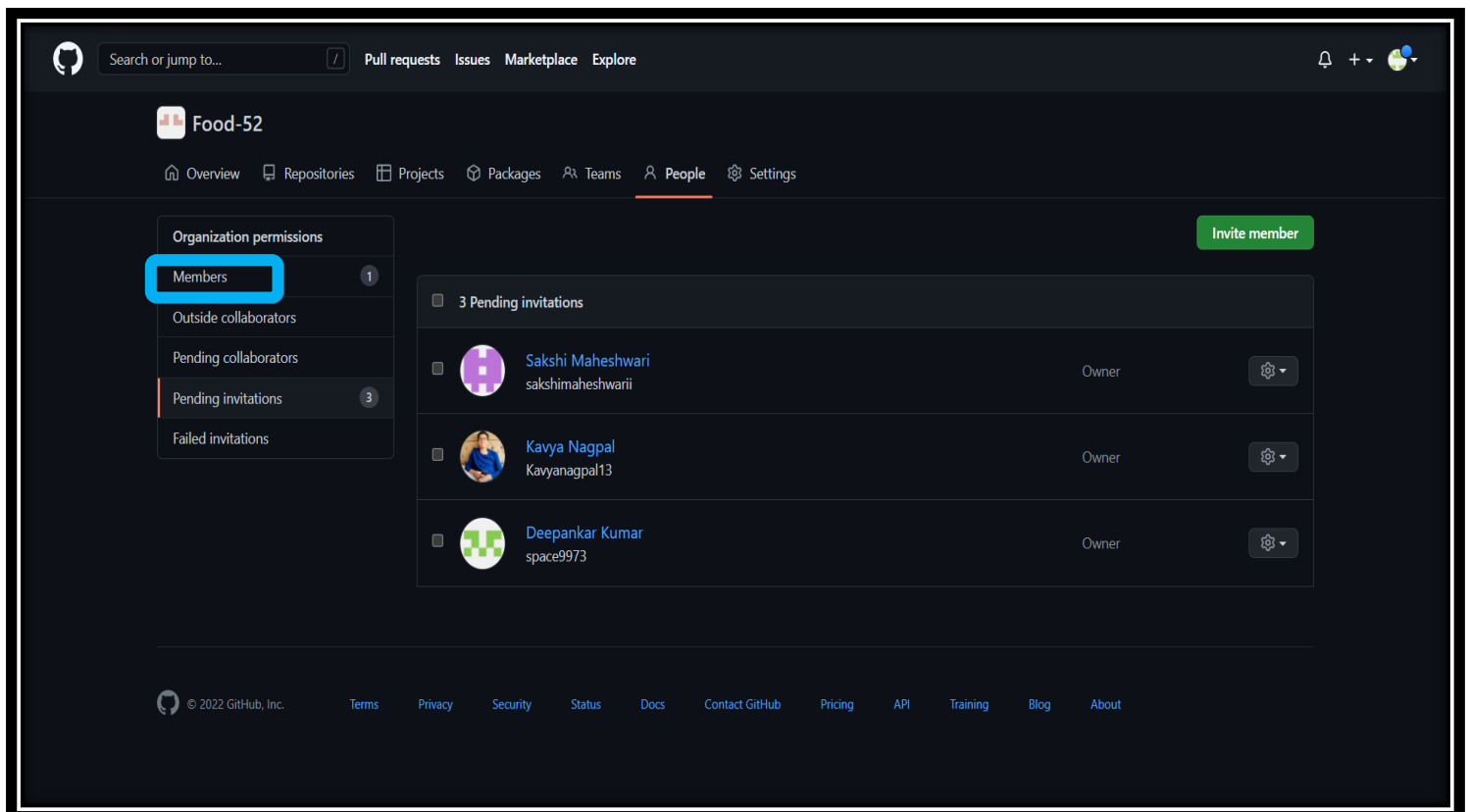
- To create a new file or upload an existing file into your repository select the option in the following box.



- 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.





Invite a member to Food-52

Search by username, full name or email address

Q sakshimaheshwarii

Invite

Invite a billing manager

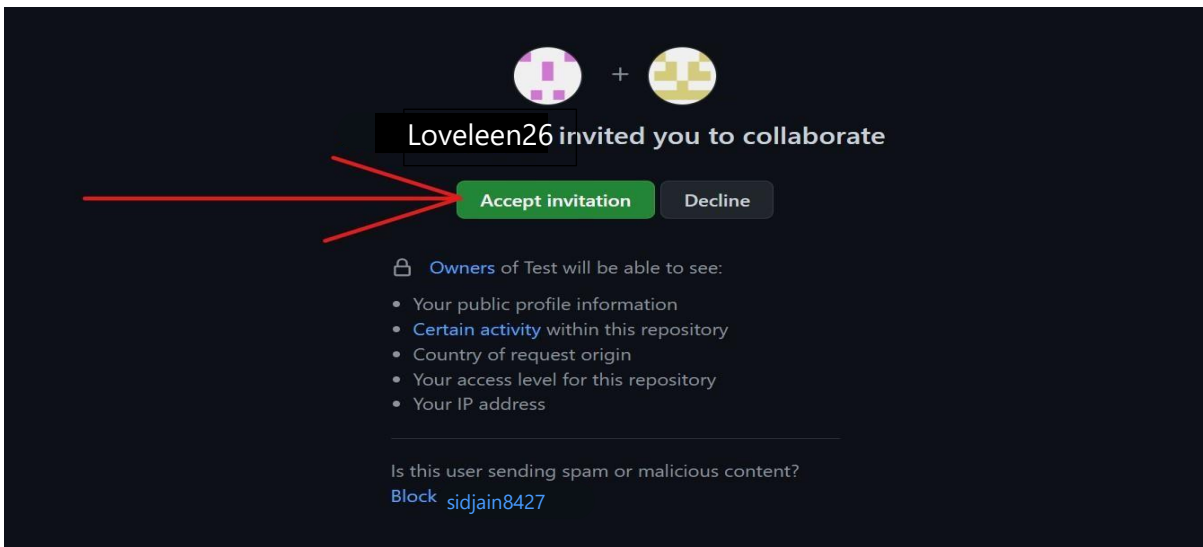


Authenticate your members with SAML single sign-on

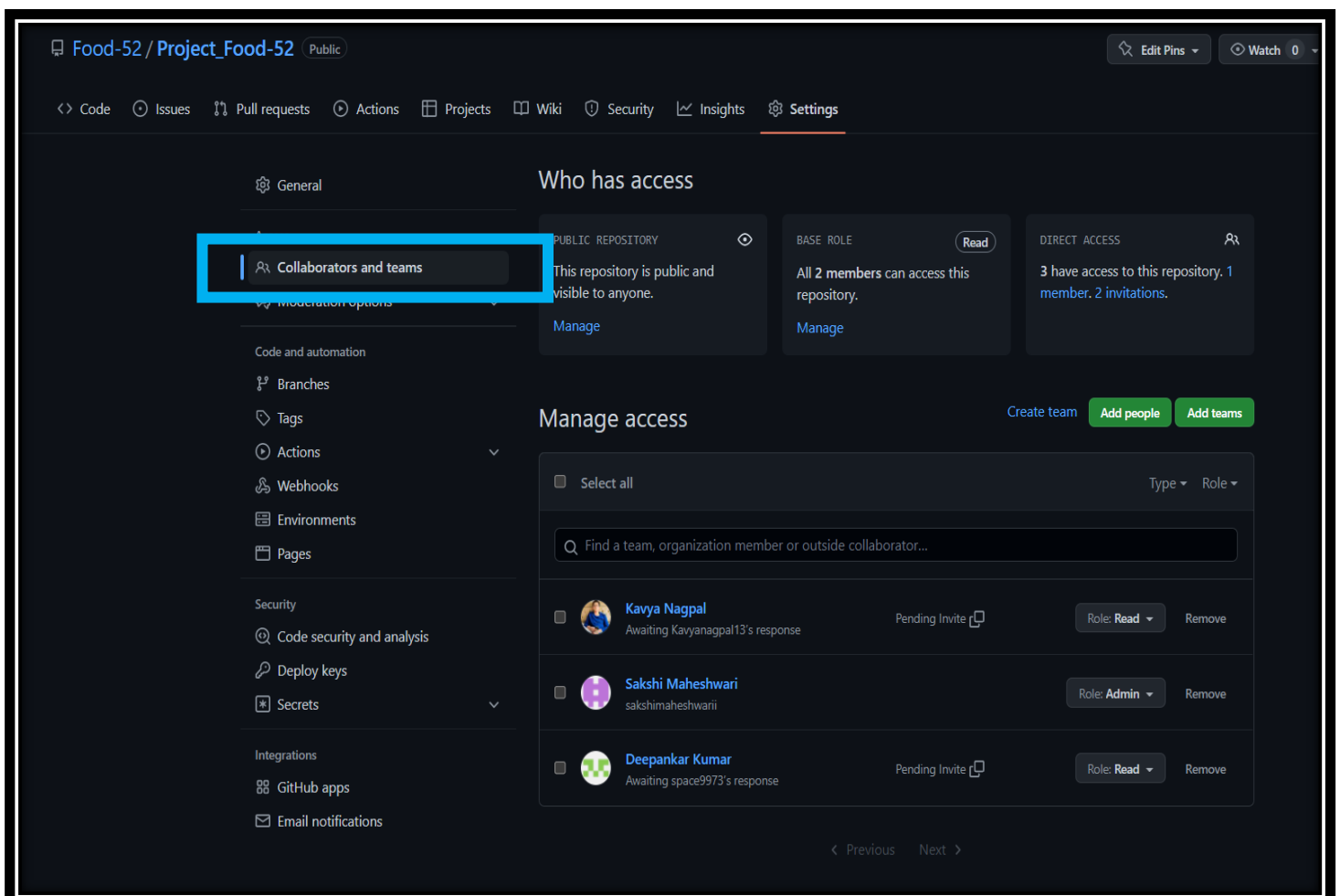
Try risk-free for 30 days, [learn more about SAML](#), or [dismiss this message](#).

- 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.



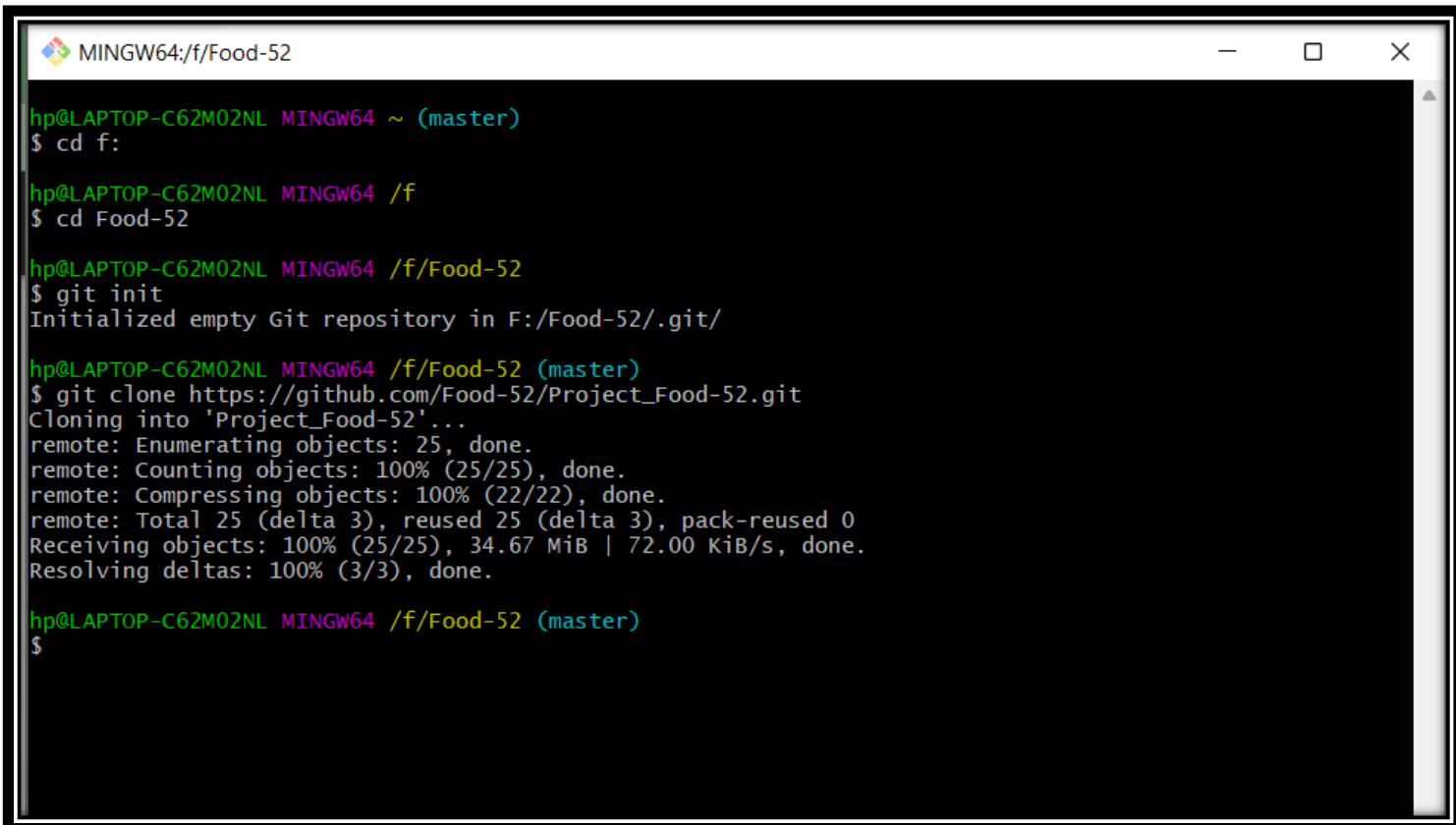
- Next, Open the desired Repository in the Organisation. Look and click on Settings -> Collaborators and Teams. Here you can Manage the role of each collaborator.



Experiment No. 02

Aim: Open and Close a Pull Request

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



```
MINGW64:/f/Food-52

hp@LAPTOP-C62M02NL MINGW64 ~ (master)
$ cd f:

hp@LAPTOP-C62M02NL MINGW64 /f
$ cd Food-52

hp@LAPTOP-C62M02NL MINGW64 /f/Food-52
$ git init
Initialized empty Git repository in F:/Food-52/.git/

hp@LAPTOP-C62M02NL MINGW64 /f/Food-52 (master)
$ git clone https://github.com/Food-52/Project_Food-52.git
Cloning into 'Project_Food-52'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 25 (delta 3), reused 25 (delta 3), pack-reused 0
Receiving objects: 100% (25/25), 34.67 MiB | 72.00 KiB/s, done.
Resolving deltas: 100% (3/3), done.

hp@LAPTOP-C62M02NL MINGW64 /f/Food-52 (master)
$
```

- Add and commit the changes to the local repository.

```
hp@LAPTOP-C62M02NL MINGW64 /f/food-52 (master)
$ git add .
warning: adding embedded git repository: Project_Food-52
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:
hint:   git submodule add <url> Project_Food-52
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached Project_Food-52
hint:
hint: See "git help submodule" for more information.

hp@LAPTOP-C62M02NL MINGW64 /f/food-52 (master)
$ ls
Project_Food-52/

hp@LAPTOP-C62M02NL MINGW64 /f/food-52 (master)
$ cd Project_Food-52

hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (main)
$ ls
1.png          PROJECT_KAVYA.html  index.html        'order now.html'   style.css
2.png          bg1.jpg            loginform.html    'payment form.html' stylepayment.css
3.png          'contact us.html'  menu.css          phone.css           text.html
'Food 52 logo.png' contact.jpg        menu.html         signup.html

hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (main)
$ git add index.html
```

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

```
hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (main)
$ git add index.html

hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (main)
$ git commit -m "changed title"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (main)
$ git remote add leap https://github.com/Food-52/Project_Food-52.git

hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (main)
$ git push leap
Everything up-to-date

hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (main)
$ |
```

```
hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (main)
$ git branch branch1.1

hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (main)
$ git checkout branch1.1
Switched to branch 'branch1.1'

hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (branch1.1)
$ touch hello.txt

hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (branch1.1)
$ git add *

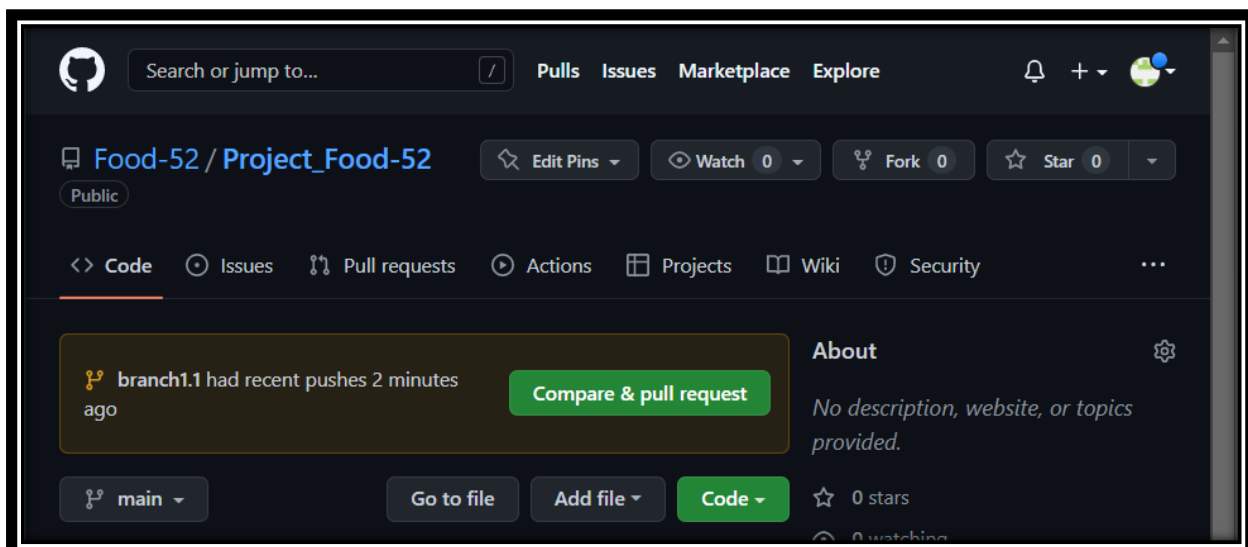
hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (branch1.1)
$ git commit -m"New file"
[branch1.1 a060419] New file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 hello.txt

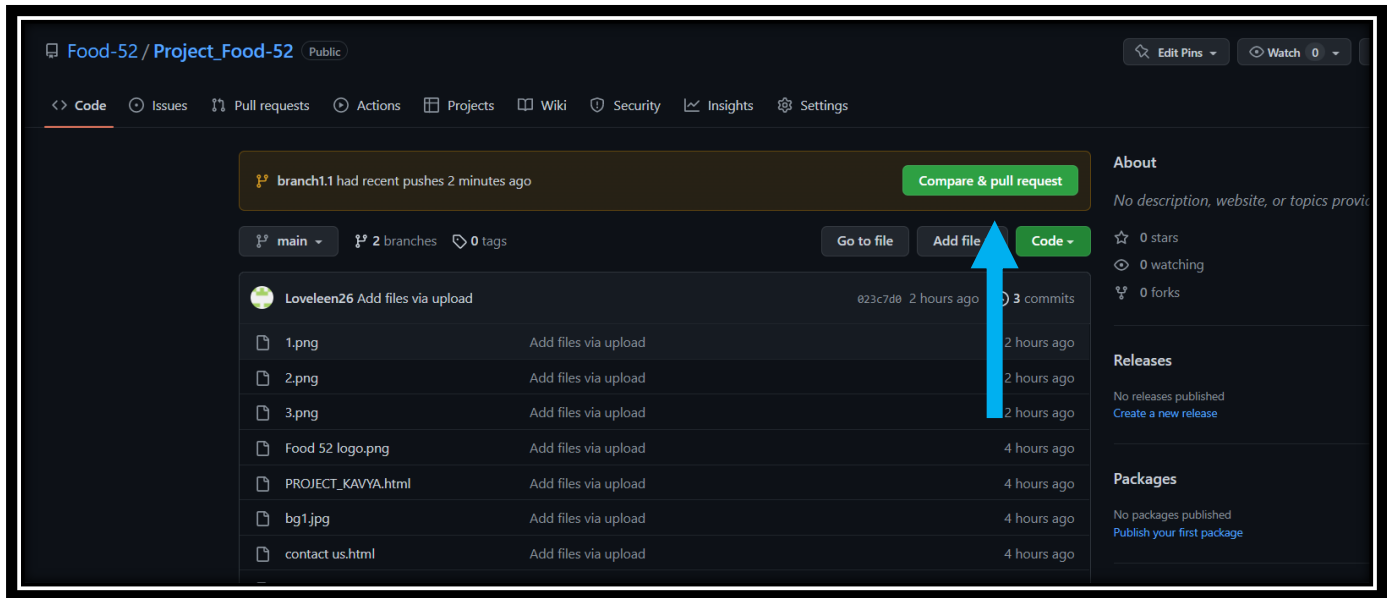
hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (branch1.1)
$ git push origin branch1.1
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 274 bytes | 274.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'branch1.1' on GitHub by visiting:
remote:   https://github.com/Food-52/Project_Food-52/pull/new/branch1.1
remote:
To https://github.com/Food-52/Project_Food-52.git
 * [new branch]      branch1.1 -> branch1.1

hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (branch1.1)
$ git remote add food https://github.com/Food-52/Project_Food-52.git

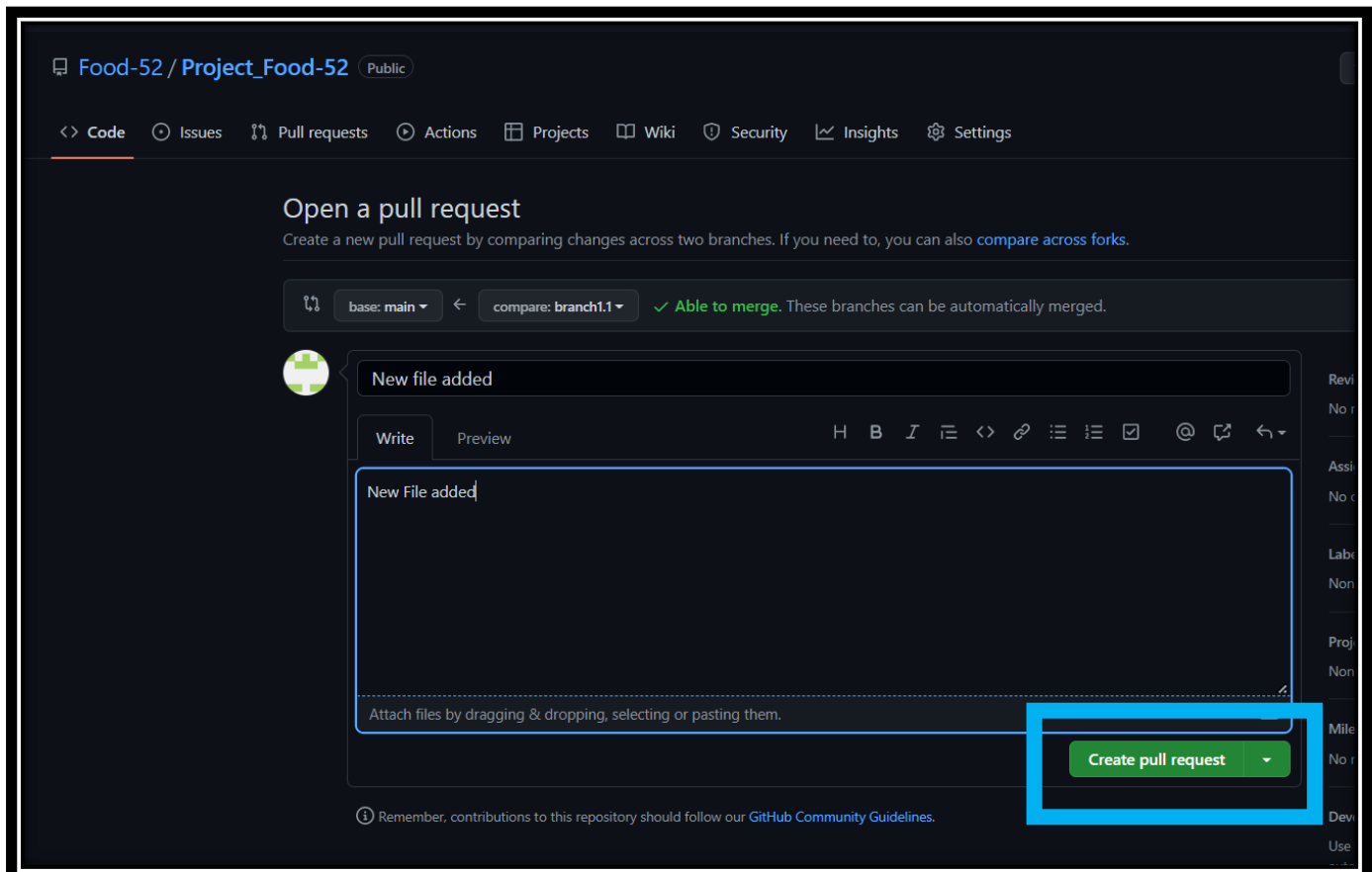
hp@LAPTOP-C62M02NL MINGW64 /f/food-52/Project_Food-52 (branch1.1)
$ git push food
Everything up-to-date
```

- 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.



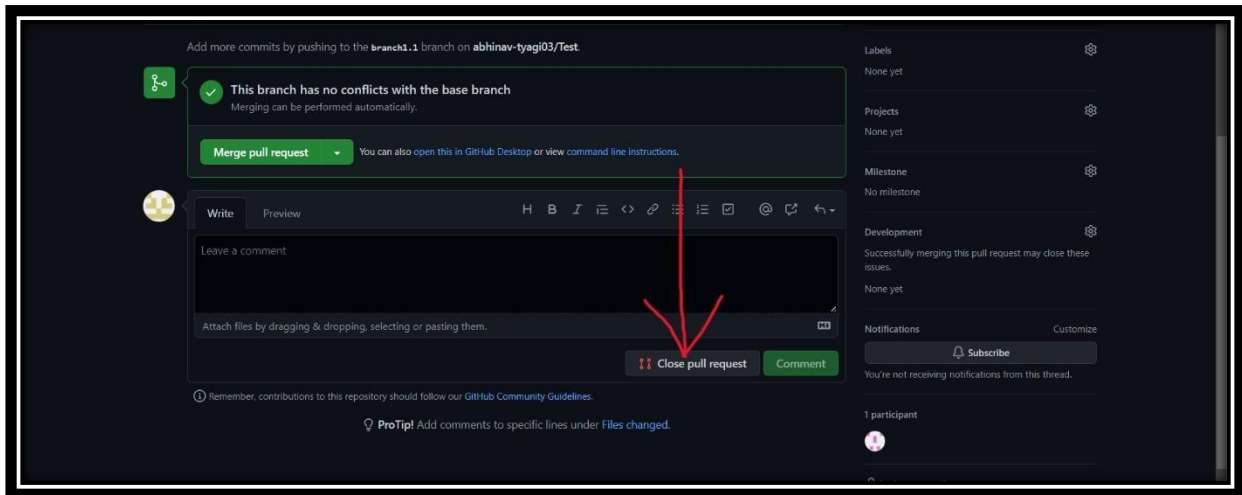
- GitHub will detect any conflicts and ask you to enter a description of your pull request.

The screenshot shows the GitHub 'Open a pull request' page. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main heading is 'Open a pull request' with a subtext: 'Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.' Below this, there's a form to select the base repository (dropdown), base branch (main), head repository (AP), and compare branch (main). A green checkmark indicates 'Able to merge. These branches can be automatically merged.' The main content area has a title field with 'changed title', a 'Write' tab, and a text area for a comment. To the right, there are settings for Reviewers, Assignees, Labels, Projects, and Milestone, all currently set to 'None yet'. At the bottom right, there's a green 'Create pull request' button. A footer note mentions the GitHub Community Guidelines.

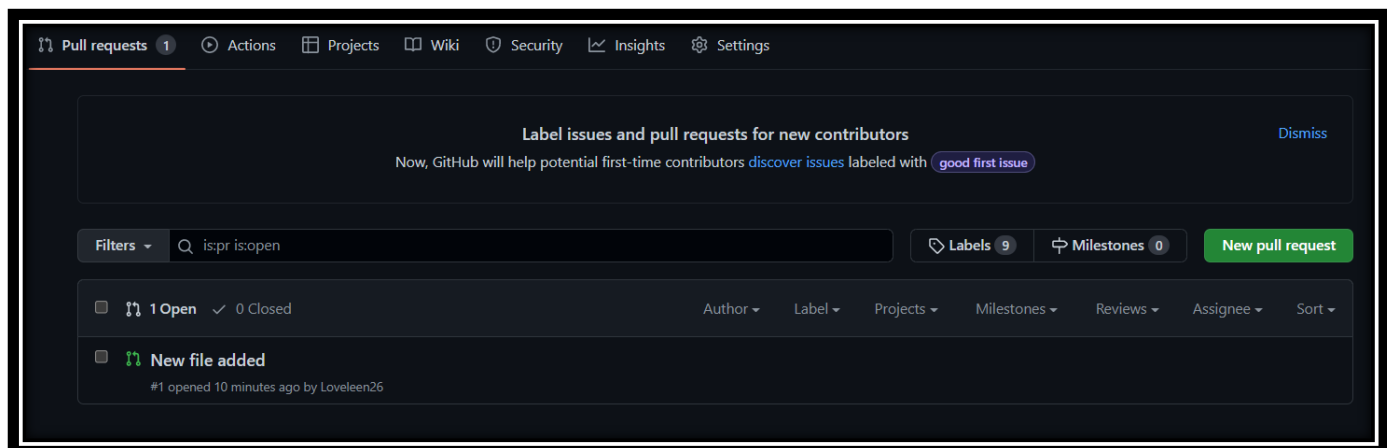
- After opening a pull request all the team members will be sent the request if they want to merge or close the request.

The screenshot shows the GitHub pull request interface after it has been opened. At the top, it says 'Add more commits by pushing to the branch1.1 branch on'. Below this, there's a green checkmark indicating 'This branch has no conflicts with the base branch' and a green 'Merge pull request' button. The main content area has a title field, a 'Write' tab, and a text area for a comment. To the right, there are settings for Labels, Projects, Milestone, and Development, all currently set to 'None yet'. At the bottom right, there's a green 'Close pull request' button and a 'Comment' button. A footer note mentions the GitHub Community Guidelines.

- 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.



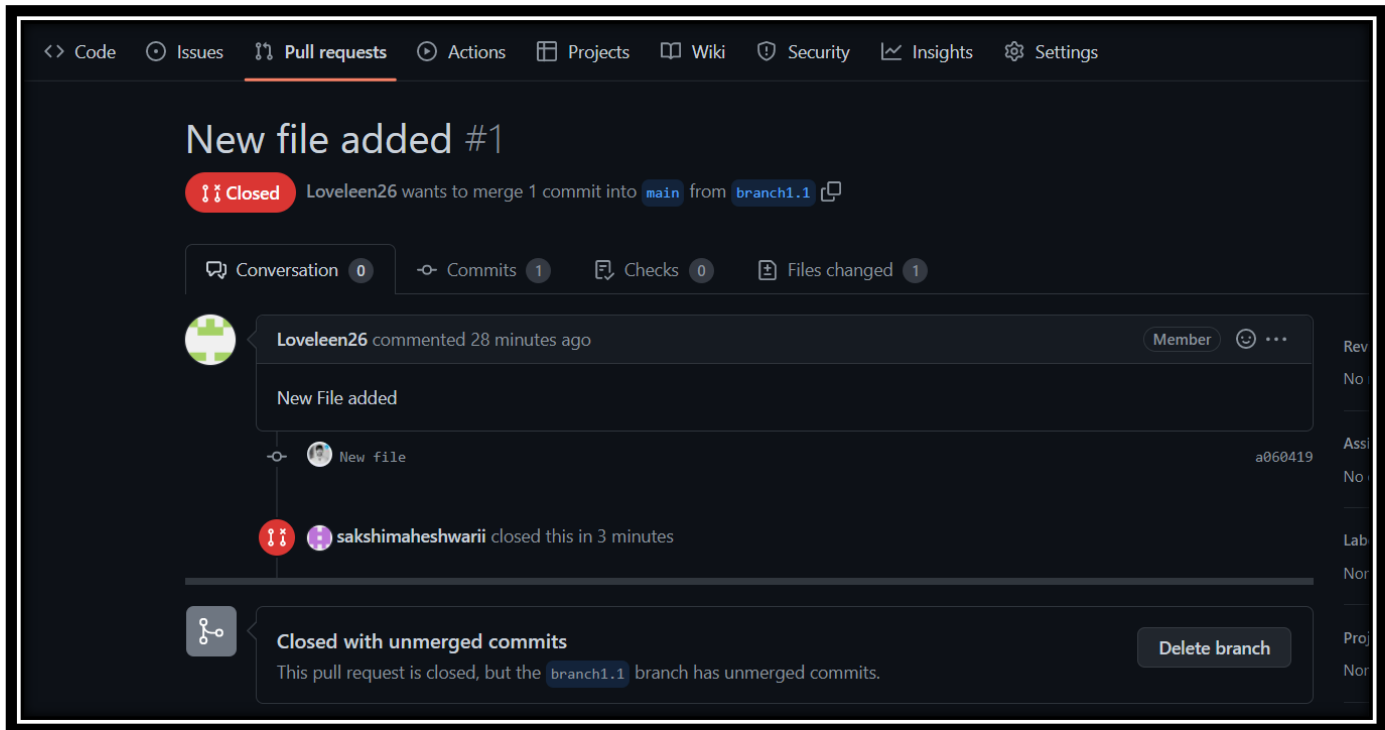
Experiment No. 03

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 options 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 .

- The result of closing the request is shown below.



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

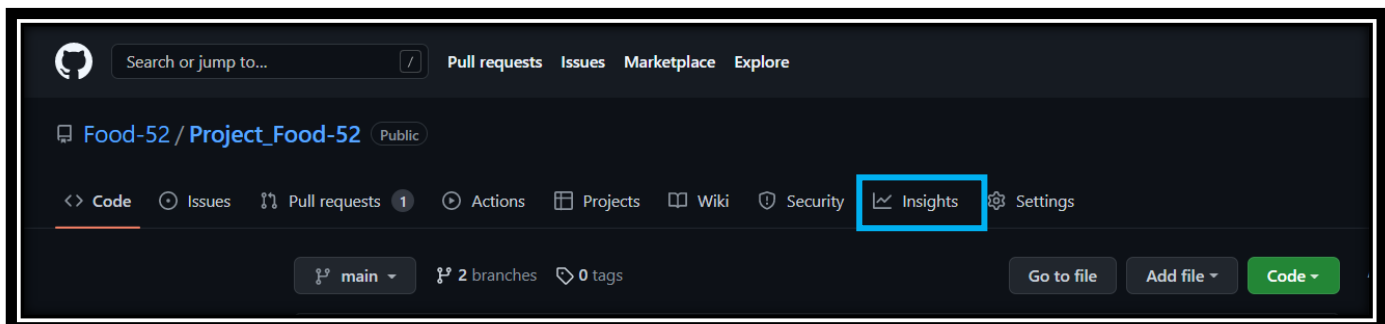
repository, you can use this data to get a better understanding of who's using your repository and why they're using it.

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

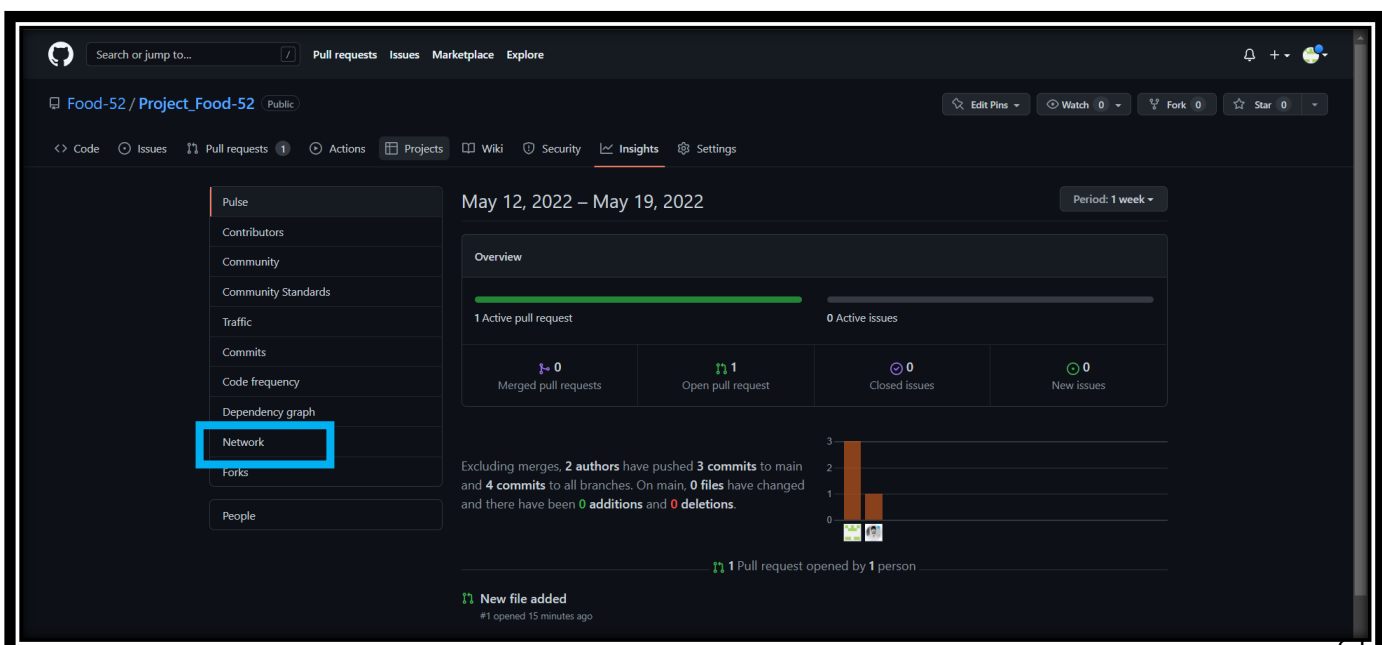
- Pulse
- Contributors
- Traffic
- Commits
- Code frequency
- Network

Steps to access network graphs of respective repository

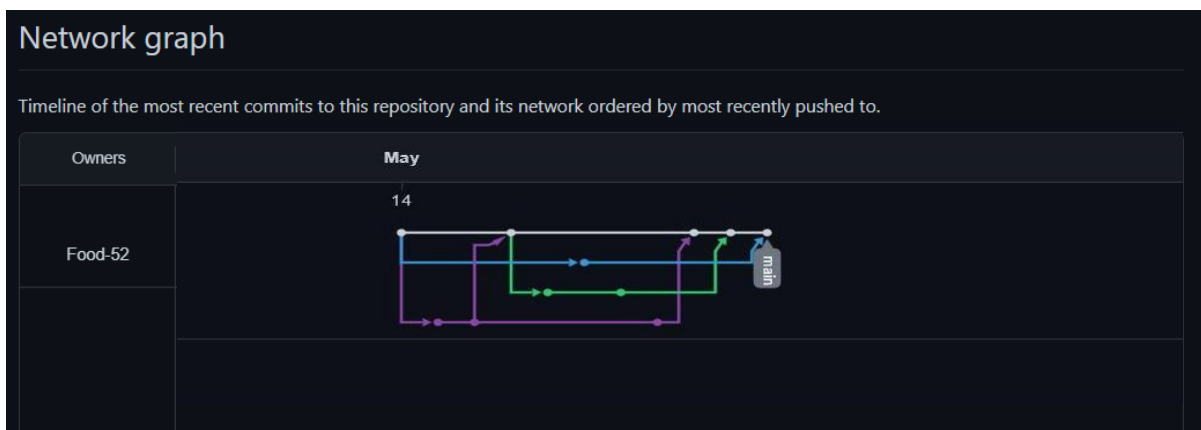
1. On GitHub.com, navigate to the main page of the repository.
2. Under your repository name, click Insights.



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.



5. Workflow and Discussion

During the development of the application we kept in mind the digital framework and financial network online. And we took an approach on how to create a infrastructure that enables cashless economy. The development of this project started with a collaboration of the team while implementing the basic framework and overview and later the project development started which was at last uploaded on Git-Hub via collaboration.