Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Zeta**

Department: **DCSE**

Submitted By:

ISHA YADAV

2110992064

G27 - B

Submitted To:

Dr. Anuj Jain

List of Tasks

S. No	Task Title	Page No.
1	Setting up of Git Client.	1-6
2	Setting up GitHub Account.	7-9
3	Generate logs.	10-11
4	Create and Visualize branches.	12-13
5	Git lifecycle description.	14-16

Task 1

Installing and Configuring the Git client

The following sections list the steps required to properly install and configure the Git clients - Git Bash and Git GUI - on a Windows 7 computer.
Git is also available for Linux and Mac. The remaining instructions here, however, are specific to the Windows installation.
Be sure to carefully follow all of the steps in the first five sections. The last section, 6, is optional.

There is also a section on common problems and possible fixes at the bottom of the document.

1. Git installation

Download the Git installation program (Windows, Mac, or Linux) from http://git-scm.com/downloads.

When running the installer, various screens appear (Windows screens shown). Generally, you can accept the default selections, ***except in the screens below where you do NOT want the default selections:***

In the **Select Components** screen, make sure **Windows Explorer Integration** is selected as shown:

Check for version :

MINGW64:/c/Users/Hp/Downloads/Git tutorials

```
Hp@LAPTOP-0R4R5FV2 MINGW64 ~/Downloads/Git tutorials
$ git --version
git version 2.35.1.windows.2

Hp@LAPTOP-0R4R5FV2 MINGW64 ~/Downloads/Git tutorials
$ git --version
git version 2.35.1.windows.2

Hp@LAPTOP-0R4R5FV2 MINGW64 ~/Downloads/Git tutorials
$ |
```

Task 2

Setting up Github account

1. Creating an account

To sign up for an account on GitHub.com, navigate to https://github.com/ and follow the prompts. To keep your GitHub account secure you should use a strong and unique password. For more information, see "Creating a strong password."

2. Choosing your GitHub product You can choose GitHub Free or GitHub Pro to get access to different features for your personal account. You can upgrade at any time if you are unsure at first which product you want. For more information on all of GitHub's plans, see "GitHub's products."

3. Verifying your email address

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

4. Configuring two-factor authentication

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

5. Viewing your GitHub profile and contribution graph

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

Screenshot :

github.com/join/recommended_plan
Learn to ship software like a pro.
GitHub gives students free access to the best developer tools so they can learn by doing.
Free
Unlimited public/private repositories
2,000 Actions minutes/month
Free for public repositories
500MB of Packages storage
Free for public repositories
Community support
Get additional student benefits
GitHub Pro
Protect your branches
Ensure that collaborators on your repository cannot make irrevocable changes to branches.
Draft pull requests
Pages and Wikis
3,000 CI/CD minutes/month
Free for public repositories
2GB of Packages storage
Free for public repositories
Web-based support
Type here to search

Task 3

Generate logs

Theory:

Git Logs:

Logs are nothing but the history which we can see in Git by using the code Git log. It contains all the past commits, insertions and deletions which can be seen anytime.

Why do we need logs?

Logs help us to check the changes made in code or files and by whom. It also contains the details of insertions and deletions and also the time it was changed at.

MINGW64:/c/Users/Hp/Downloads/Git tutorials/priyanshu

```
Hp@LAPTOP-OR4R5FV2 MINGW64 ~/Downloads/Git tutorials
$ cd priyanshu

Hp@LAPTOP-OR4R5FV2 MINGW64 ~/Downloads/Git tutorials/priyanshu (main)
$ git log
commit 24d9d97284a50e1050d218b7e90439bb4ffb4121 (HEAD -> main, origin/main, orig
in/HEAD)
Author: priyanshu996 <102685453+priyanshu996@users.noreply.github.com>
Date:   Thu Mar 31 15:26:51 2022 +0530

    Add files via upload

Hp@LAPTOP-OR4R5FV2 MINGW64 ~/Downloads/Git tutorials/priyanshu (main)
$
```

Task 4

Create and Visualize Branches

Theory:

How to create branches?

The main branch in git is called the master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the files which are created in branch are not shown in master branch. We can also merge both the parent (master) and child (other branches).

1. For creating a new branch: git branch “name of the branch”

2. To check how many branches we have: git branch

MINGW64:/c/Users/Hp/Downloads/Git tutorials/priyanshu

```
Hp@LAPTOP-OR4R5FV2 MINGW64 ~/Downloads/Git tutorials
$ cd priyanshu

Hp@LAPTOP-OR4R5FV2 MINGW64 ~/Downloads/Git tutorials/priyanshu (main)
$ git log
commit 24d9d97284a50e1050d218b7e90439bb4ffb4121 (HEAD -> main, origin/main, orig
in/HEAD)
Author: priyanshu996 <102685453+priyanshu996@users.noreply.github.com>
Date:   Thu Mar 31 15:26:51 2022 +0530

    Add files via upload

Hp@LAPTOP-OR4R5FV2 MINGW64 ~/Downloads/Git tutorials/priyanshu (main)
$ git branch yoo

Hp@LAPTOP-OR4R5FV2 MINGW64 ~/Downloads/Git tutorials/priyanshu (main)
$ yoo
bash: yoo: command not found

Hp@LAPTOP-OR4R5FV2 MINGW64 ~/Downloads/Git tutorials/priyanshu (main)
$ pwd
/c/Users/Hp/Downloads/Git tutorials/priyanshu

Hp@LAPTOP-OR4R5FV2 MINGW64 ~/Downloads/Git tutorials/priyanshu (main)
$ git branch yoo
fatal: a branch named 'yoo' already exists

Hp@LAPTOP-OR4R5FV2 MINGW64 ~/Downloads/Git tutorials/priyanshu (main)
$ |
```

Task 5

Git Lifecycle Description

Theory:

Stages in GIT Life Cycle:

Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, when a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states:

- Working directory
- Staging area
- Git directory

Working Directory:

Consider a project residing in your local system. This project may or may not be tracked by Git. In either case, this project directory is called your Working directory.

Staging Area:

Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.

Git Directory:

Now that the files to be committed are grouped and ready in the staging area, we can commit these files. So, we commit this group of files along with a commit message explaining what is the commit about. Apart from commit message, this step also records the author and time of the commit. Now, a snapshot of the files in the commit is recorded by Git. The information related to this commit is stored in the Git directory.

Remote Repository: means mirror or clone of the local Git repository in GitHub. And pushing means uploading the commits from local Git repository to remote repository hosted in GitHub.

Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Zeta**

Department: **DCSE**

Submitted By: Isha Yadav **Submitted To:** Mr. Anuj Jain
2110992064
G-27

List of Tasks

S. No	Task Title	Page No.
1	Add Collaborators on GitHub Repo.	3 -6
2	Fort And Commit.	7 -10
3	Merge and resolve conflicts.	11-14
4	Reset and Revert.	15 -18

Task-1

ADDING COLLABORATORS ON GITHUB REPO.

CREATING A REPOSITORY

1. Click the new repository button in the top-right. You'll have an option there to initialize the repository with a README file, but I don't. 2. Click the “Create repository” button.

Inviting collaborators to a personal repository

1. Ask for the username of the person you're inviting as a collaborator.
2. On your Github, navigate to the main page of the repository.

3. Under your repository name, click **Settings**.

4. In the left sidebar, click **Collaborators**.

5. Under "Collaborators", start typing the collaborator's username.

Manage access

github.com/ishayadav3499/task-2/settings/access

Code and automation
- Branches
- Tags
- Actions
- Webhooks
- Environments
- Pages

Security
- Code security and analysis
- Deploy keys
- Secrets

Integrations
- GitHub apps
- Email notifications

Manage

to this repository.

Manage access

You haven't invited any collaborators yet

Add people

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

Activate Windows
Go to Settings to activate Windows.

Type here to search 43°C Haze ENG 10:45 29-04-2022

Manage access

github.com/ishayadav3499/task-2/settings/access

Code and automation
- Branches
- Tags
- Actions
- Webhooks
- Environments
- Pages

Security
- Code security and analysis
- Deploy keys
- Secrets

Integrations
- GitHub apps
- Email notifications

Manage

to this repository.

Manag

Add a collaborator to **task-2**

Khushisaharan

khushisaharan
Invite collaborator

Select a collaborator above

ors yet

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

Activate Windows
Go to Settings to activate Windows.

Type here to search 43°C Haze ENG 10:45 29-04-2022

6. Select the collaborator's username from the drop-down menu.

7. Click Add Collaborator

Task-2

FORK AND COMMIT

1. Fork a GitHub repository.

2. Clone the forked repository to your local system

3. Add a git remote for the original repository.

4. Create a feature branch in which to place your changes.

5. Make your changes to the new branch.

6. Commit the changes to the branch.

7. Push the branch to GitHub

MINGW64:/c/Users/DELL/Desktop/task2/task-2

```
DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/task2 (master)
$ ls
task-2/

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/task2 (master)
$ cd task2
bash: cd: task2: No such file or directory

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/task2 (master)
$ git add
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'?
hint: Turn this message off by running
hint: "git config advice.addEmptyPathspec false"

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/task2 (master)
$ ls
task-2/

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/task2 (master)
$ cd task-2

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/task2/task-2 (main)
$ ls
 README.md   bubble_sort.cpp             fahrenheit_to_celcius.cpp  'greater of three.cpp'      hollow_rectangle_pattern.cpp   linear_search.cpp
 array.cpp  'factorail_of_n number.cpp'  'fibonacci series.cpp'       half_pyramid_pattern.cpp   insertion_sort.cpp

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/task2/task-2 (main)
$ git add
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'?
hint: Turn this message off by running
hint: "git config advice.addEmptyPathspec false"

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/task2/task-2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/task2/task-2 (main)
$ git commit -m "add code"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/task2/task-2 (main)
$
```

Activate Windows
Go to Settings to activate Windows.

Type here to search
43°C ENG 10:56 29-04-2022

C

ishayadav3499/task-2

github.com/ishayadav3499/task-2

Search or jump to... / Pull requests Issues Marketplace Explore

ishayadav3499 / task-2 Public

Pin Watch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

ishayadav3499 Add files via upload 1b0ef8f 33 minutes ago 2 commits

README.md	Initial commit	37 minutes ago
array.cpp	Add files via upload	33 minutes ago
bubble_sort.cpp	Add files via upload	33 minutes ago
factorail_of_n number.cpp	Add files via upload	33 minutes ago
fahrenheit_to_celcius.cpp	Add files via upload	33 minutes ago
fibonacci series.cpp	Add files via upload	33 minutes ago
greater of three.cpp	Add files via upload	33 minutes ago
half_pyramid_pattern.cpp	Add files via upload	33 minutes ago
hollow_rectangle_pattern.cpp	Add files via upload	33 minutes ago
insertion sort.cpp	Add files via upload	33 minutes ago

About

No description, website, or topics provided.

Readme

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Activate Windows

Go to Settings to activate Windows.

Type here to search

43°C Haze ENG 10:47 29-04-2022

TASK 3

Merging and Resolving conflicts.

Merge conflicts

So things can go wrong, which usually starts with a **merge conflict**, due to both collaborators making incompatible changes to a file. While the error messages from merge conflicts can be daunting, getting things back to a normal state can be

straightforward once you've got an idea where the problem lies.

A merge conflict occurs when both the owner and collaborator change the same lines in the same file without first pulling the changes that the other has made. This is most easily avoided by good communication about who is working on various sections of each file, and trying to avoid overlaps. But sometimes it happens, and *git* is there to warn you about potential problems. And git will not allow you to overwrite one person's changes to a file with another's changes to the same file if they were based on the same version.

```
# Start a new feature
git checkout -b new-feature main
# Edit some files
git add <file>
git commit -m "Start a feature"
# Edit some files
git add <file>
git commit -m "Finish a feature"
# Merge in the new-feature branch
git checkout main
git merge new-feature
git branch -d new-feature
```

How to resolve a conflict

Abort, abort, abort…

Sometimes you just made a mistake. When you get a merge conflict, the repository is placed in a 'Merging' state until you resolve it. There's a commandline command to abort abort doing the merge altogether:

```
git merge -abort
```

To see the conflict, you can go back to the changes section. Here, you'll see the files that contain conflicts. In this case, it is the `README.md` file.

Resolving conflicts

The easiest way to resolve a conflict is to change the file on your computer. If you open `README.md` now, you'll see lines that say this:

```
<<<<<< HEAD Hello world! This is my third Github repo!====== Hello world!
```

```
This is my second Github repo! >>>&gt;>> snt2h1s3n4tnthd9au8d3324
```

```
On branch main
Unmerged paths:
(use "git add/rm ..." as appropriate to mark resolution)
both modified: hello.py
```

The code between <<<;<<; HEAD and ======= is the code in our local repository (our code).

The code between ====== and >>t;>>>> is the code from the remote repository (their code).

These two lines of code are conflicting. We need to choose between the “second Github repo” or the “third Github repo”. To fix the conflict, you choose the correct line of code. Then you delete everything else.

In this case, let’s say “third” is the correct version. What you’ll do is delete everything else that’s incorrect.

```
Hello world! This is my third Github repo!
```
Committing the merge

When you head back to Fork, you’ll see that the changes are updated. In our case, the change is the one on our local, so we don’t see any files that need to be staged.

If the change is different, you will need to stage the files.
After staging, you need to commit the merge.

If you look at the commit message area, you'll see that Fork has filled in a commit message for you automatically. You can use the commit message directly.

Click on commit to commit the changes

TASK 4

RESET AND REVERT

GIT RESET

The `git reset` command allows you to RESET your current head to a specified state. You can reset the state of specific files as well as an entire branch. This is useful if you haven't pushed your commit up to GitHub or another remote repository yet.

Reset a file or set of files

The following command lets you selectively choose chunks of content and revert or unstage it.

```
git reset (--patch | -p) [tree-ish] [--] [paths]
```

Unstage a file

If you moved a file into the staging area with `git add`, but no longer want it to be part of a commit, you can use `git reset` to unstage that file:

```
git reset HEAD FILE-TO-UNSTAGE
```

The changes you made will still be in the file, this command just removes that file from your staging area.

Reset a branch to a prior commit

The following command resets your current branch's HEAD to the given `COMMIT` and updates the index. It basically rewinds the state of your branch, then all commits you make going forward write over anything that came after the reset point. If you omit the `MODE`, it defaults to `--mixed`:

```
git reset MODE COMMIT
```

The options for `MODE` are:

- `--soft`: does not reset the index file or working tree, but resets HEAD to `commit`. Changes all files to "Changes to be commited"
- `--mixed`: resets the index but not the working tree and reports what has not been updated
- `--hard`: resets the index and working tree. Any changes to tracked files in the working tree since `commit` are discarded
- `--merge`: resets the index and updates the files in the working tree that are different between `commit`and HEAD, but keeps those which are different between the index and working tree
- `--keep`: resets index entries and updates files in the working tree that are different between `commit`and HEAD. If a file that is different between `commit` and HEAD has local changes, the reset is aborted

GIT REVERT

Git Revert

Both the `git revert` and `git reset` commands undo previous commits. But if you've already pushed your commit to a remote repository, it is recommended that you do not use `git reset` since it rewrites the history of commits. This can make working on a repository with other developers and maintaining a consistent history of commits very difficult. Instead, it is better to use `git revert`, which undoes the changes made by a previous commit by creating an entirely new commit, all without altering the history of commits.

Revert a commit or set of commits

The following command lets you revert changes from a previous commit or commits and create a new commit.

```
git revert [--[no-]edit] [-n] [-m parent-number] [-s] [-S[<keyid>]] <commit>… git
revert --continue
```

```
git revert --quit git
revert –abort
```

```
C:\Users\ACER\Desktop\Git>git log
commit 91ea2001532d811a686f7f6520c845730b4499fa (HEAD -> master)
Author: pikachua7 <atharvapaliwal7@gmail.com>
Date:   Thu Jan 27 23:44:08 2022 +0530

    Revert "Added Hello World"

    This reverts commit 63336594be4dec60bac479e8f028fa9db3bbc3ce.

commit 63336594be4dec60bac479e8f028fa9db3bbc3ce (origin/master)
Author: pikachua7 <atharvapaliwal7@gmail.com>
Date:   Thu Jan 27 23:43:11 2022 +0530

    Added Hello World

commit fb545a2477e23c878dcbf7a64ce53f31e60ed2a5
Author: pikachua7 <atharvapaliwal7@gmail.com>
Date:   Thu Jan 27 23:42:29 2022 +0530

    Added Hello Geeks
```

Common options:

`-e`

`--edit`

- This is the default option and doesn't need to be explicitly set. It opens your system's default text editor and lets you edit the new commit message before commit the revert.
- This option does the opposite of `-e`, and `git revert` will not open the text editor.
- This option prevents `git revert` from undoing a previous commit and creating a new one. Rather than creating a new commit, `-n` will undo the changes from the previous commit and add them to the Staging Index and Working Directory.

`--no-edit`

`-n`

`-no-commit`

A Project report on

The coffee shop

with

Source Code Management

(CS181)

Submitted by

Team Member 1	**Khushi**	**2110992029**
Team Member 2	**Isha**	**2110992064**
Team Member 3	**Vanshika**	**2110992051**
Team Member 4	**Prakriti**	**2110992053**

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**		

Submitted by:

Name: Isha Yadav

Table of Content

S. No.	Title	Page No.
1	Version control with Git	3-5
2	Problem Statement	5
3	Objective	5,6
4	Resources Requirements – Frontend / Backend	6
5	Concepts and commands, Workflow	7-22
6.	Snapshots of Project	23-25

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.

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This tutorial teaches you GitHub essentials like repositories, branches, commits, and pull requests.

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.

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

One common management-related problem faced by coffee shop owners is slow service, especially in establishments that offer many customized coffee preparations. The best POS systems also make management hassle-free by tracking customer information, using traffic patterns to forecast staffing needs, adjusting pricing, generating reports that contain actionable information, facilitating tip reconciliation, and integrating with other solutions, such as accounting software.

3. Objective

PROJECT NAME: COFFEE SHOP

The objective of this project is to provide a coffee shop that will not only be known for serving great coffee but also an environment to relax, socialize, and network. This goals of this project will be:

- To lease, renovate and open an environmentally friendly coffeeshop that will attract customers. The shop will have a low economic footprint and consider the environment as the first priority of the business
- To open with-in six months at a cost not to exceed 3% of the budgeted $400,000.
- Design a seating capacity for 30 people all with available outlets for electronic devices and shop wide Wi Fi service
- We will be offering many different types of coffee from all around the world
- Employ a volunteer program in which customers can pay for their order with lab or. This will ensure that each and every person is accommodated and has an opportunity to enjoy coffee and food in our shop regardless of the amount they have.

4. Resources Required.

Frontend – HTML5, CSS3, Java script

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.

- 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.

Search or jump to...
Pull requests
Issues
Marketplace
Explore
Khushi-Coffee
Overview
Repositories
Projects
Packages
Teams
People
Settings
Organization permissions
Members
2
Outside collaborators
Pending collaborators
Pending invitations
2
Failed invitations
Find a member...
Export
Invite member
Members
2FA
Role
ishayadav3499
2FA
Private
Member
0 teams
khushisaharan
2FA
Private
Owner
0 teams
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

- 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:/c/CoffeeShop

```
HP@LAPTOP-R9TNMH35 MINGW64 ~
$ cd c:

HP@LAPTOP-R9TNMH35 MINGW64 /c
$ mkdir CoffeeShop

HP@LAPTOP-R9TNMH35 MINGW64 /c
$ cd CoffeeShop

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop
$ git init
Initialized empty Git repository in C:/CoffeeShop/.git/

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop (master)
$ git clone https://github.com/Khushi-Coffee/MyCoffeeShop.git
Cloning into 'MyCoffeeShop'...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 33 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (33/33), 1.49 MiB | 2.29 MiB/s, done.

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop (master)
$
```

- Add and commit the changes to the local repository.

```
HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop (master)
$ git add .
warning: adding embedded git repository: MyCoffeeShop
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> MyCoffeeShop
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached MyCoffeeShop
hint:
hint: See "git help submodule" for more information.

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop (master)
$ ls
MyCoffeeShop/

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop (master)
$ cd MyCoffeeShop

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (main)
$ ls
 about-img.jpeg   blog-3.jpeg       cart-item-3.png      home-img.jpeg   index.js     menu-2.png   menu-5.png   modules.xml   pic-3.png       product-3.png
 blog-1.jpeg      cart-item-1.png   cart-item-4.png      index.css       logo.png     menu-3.png   menu-6.png   pic-1.png     product-1.png   quote-img.png
 blog-2.jpeg      cart-item-2.png   'coffee website.iml' index.html      menu-1.png   menu-4.png   misc.xml     pic-2.png     product-2.png   workspace.xml

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (main)
$ git add *

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (main)
$ git add index.html
```

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

```
HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (main)
$ git add index.html

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (main)
$ git branch Cafena

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (main)
$ git checkout Cafena
Switched to branch 'Cafena'

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (Cafena)
$ touch coffeeDate.txt

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (Cafena)
$ git add *

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (Cafena)
$ git commit -m"New File"
[Cafena cea4be0] New File
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 coffeeDate.txt

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (Cafena)
$ git push origin Cafena
```

```
HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (main)
$ git branch Cafena

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (main)
$ git checkout Cafena
Switched to branch 'Cafena'

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (Cafena)
$ touch coffeeDate.txt

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (Cafena)
$ git add *

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (Cafena)
$ git commit -m"New File"
[Cafena cea4be0] New File
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 coffeeDate.txt

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (Cafena)
$ git push origin Cafena
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 278 bytes | 139.00 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 'Cafena' on GitHub by visiting:
remote:      https://github.com/Khushi-Coffee/MyCoffeeShop/pull/new/Cafena
remote:
To https://github.com/Khushi-Coffee/MyCoffeeShop.git
 * [new branch]      Cafena -> Cafena

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (Cafena)
$ git remote add Coffee https://github.com/Khushi-Coffee/MyCoffeeShop.git

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (Cafena)
$ git push Coffee
Everything up-to-date

HP@LAPTOP-R9TNMH35 MINGW64 /c/CoffeeShop/MyCoffeeShop (Cafena)
$
```

- 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 branch name.
- 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 Git hub 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 .

- 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

<u>Steps to access network graphs of respective repository</u>

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.

Home
About
Menu
Products
Review
Contact
Blogs
FRESH COFFEE IN THE MORNING
Good Communication Is Just As Stimulating As Black Coffee, And Just As Hard To Sleep After.
Get Yours Now

Home
About
Menu
Products
Review
Contact
Blogs
ABOUT US
What Makes Our Coffee Special?
The Pain Itself Is Important To The Main Adipisicing Elite. Are There Any Pleasures In It, For The Flight Time Itself, Which Is The Nature Of The Other Services, And That's The Chronological Order? It Is No Easy Dream. From, Or?
The Coffee Quality Is Beyond The Final Product Served On Your Cup. Its Quality Springs From The Start—The Type Of Coffee Beans Grown, The Region, Roasting, And Brewing, Down To The Actual Coffee Served Right Under Your Nose.
Learn More
OUR MENU

Home About Menu Products Review Contact Blogs
CUSTOMER'S REVIEW
I Like This Coffee It's Really Smooth And Has An After Taste Of Chocolate. Service Was Brilliant And The Coffee Was Delivered The Day After I Ordered It And Was Ground Perfectly For The Cafetiere.
Ayush Aggarwal
This Is One Of Our Favourites And The Love Of Coffee Delivered It In Time For My Saturday Am Coffee
Riya Singh
This Coffee Has A Bit Of Sweetness Without Compromising The Taste. Good For Afternoon When You Want Something Sweeter
Rhythm

Home About Menu Products Review Contact Blogs
OUR PRODUCTS
NICARAGUA
COLUMBIA
PERU
COFFEE
Fresh Coffee
115rs 215rs
Fresh Coffee
115rs 150
Fresh Coffee
120rs150

Home
About
Menu
Products
Review
Contact
Blogs
OUR BLOGS
Tasty And Refreshing Coffee
By Admin / 21st May, 2021
Read More
Tasty And Refreshing Coffee
By Admin / 21st May, 2021
Eating Rice Cakes Is Like Chewing On A Foam Coffee Cup, Only Less Filling.
Read More
Tasty And Refreshing Coffee
By Admin / 21st May, 2021
A 41-Inch Bust And A Lot Of Perseverance Will Get You More Than A Cup Of Coffee - A Lot More.
Read More

Home
About
Menu
Products
Review
Contact
Blogs
CONTACT US
GET IN TOUCH
name
email
number
Contact Now