TASK 1.2

SOURCE CODE MANAGEMENT

(CS-181)

Submitted By:
Abhijit Singh
Roll no.-2110990041

Submitted To:
Dr. Monit Kapoor

Experiment 1

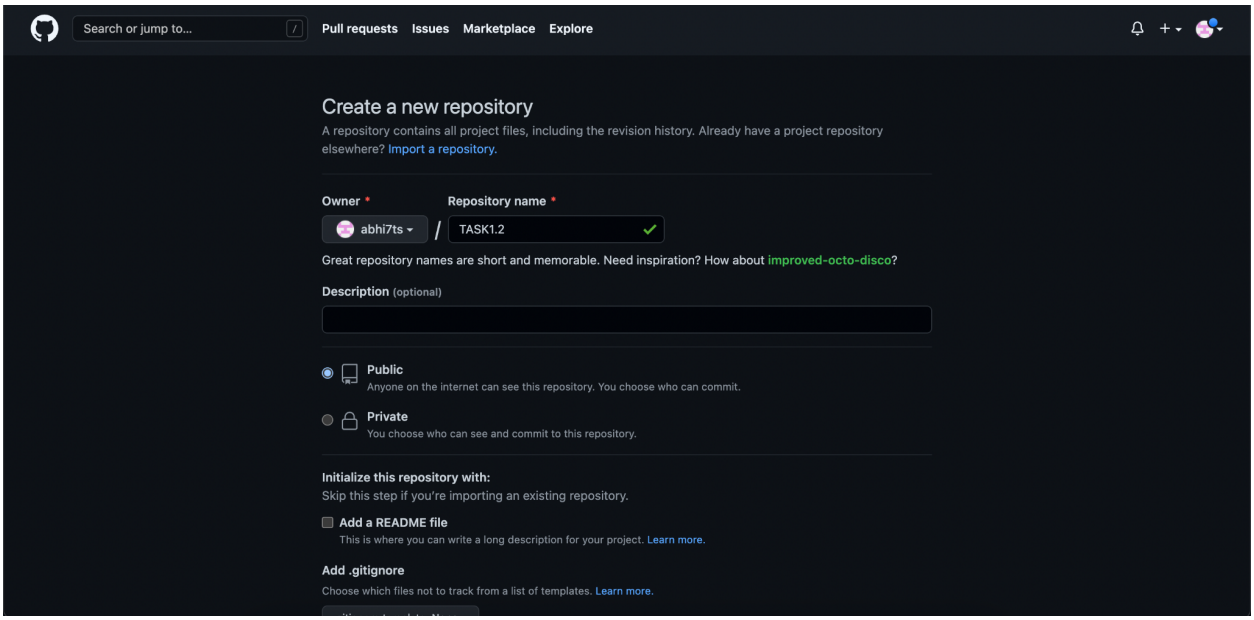
AIM: Add collaborators on GitHub repo.

THEORY:

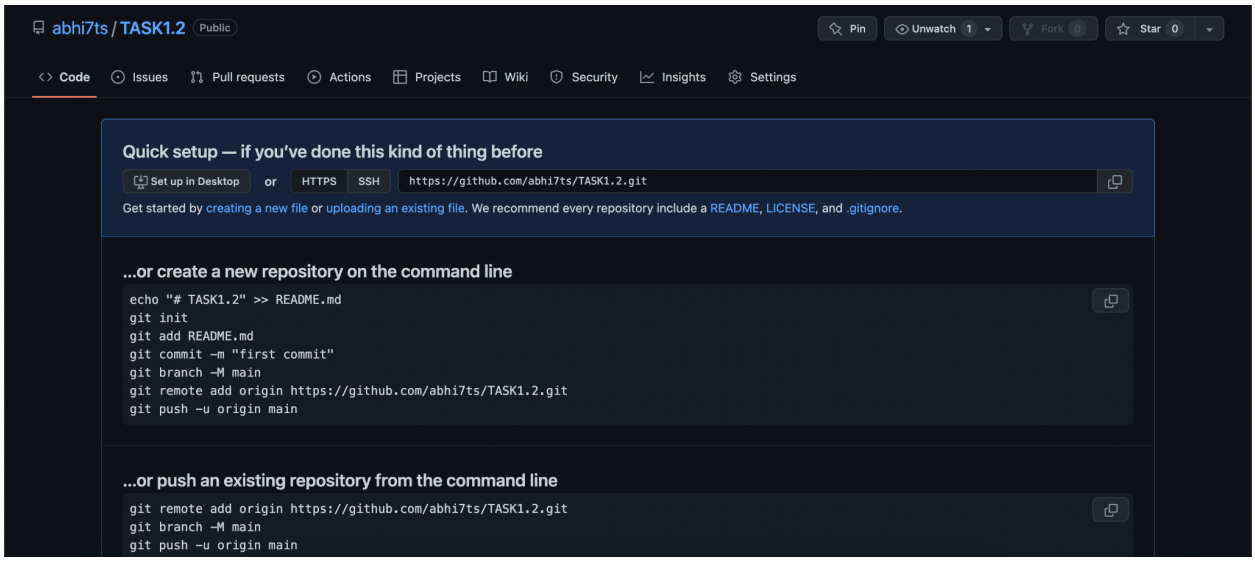
- Whenever you make a repository in GitHub, not everyone has the permission to change or push codes into your repository. The users have read-only access. In order to allow other individuals to make changes to your repository, you need to invite them to collaborate on the project.
- GitHub also restricts the number of collaborators we can invite within a period of 24 hours. If we exceed the limit, then either we have to wait for 24-hours or we can also create an organization to collaborate with more people.
- Being a collaborator, the user can create, merge and close pull requests in the repository. They can also remove them as collaborators.

PROCEDURE:

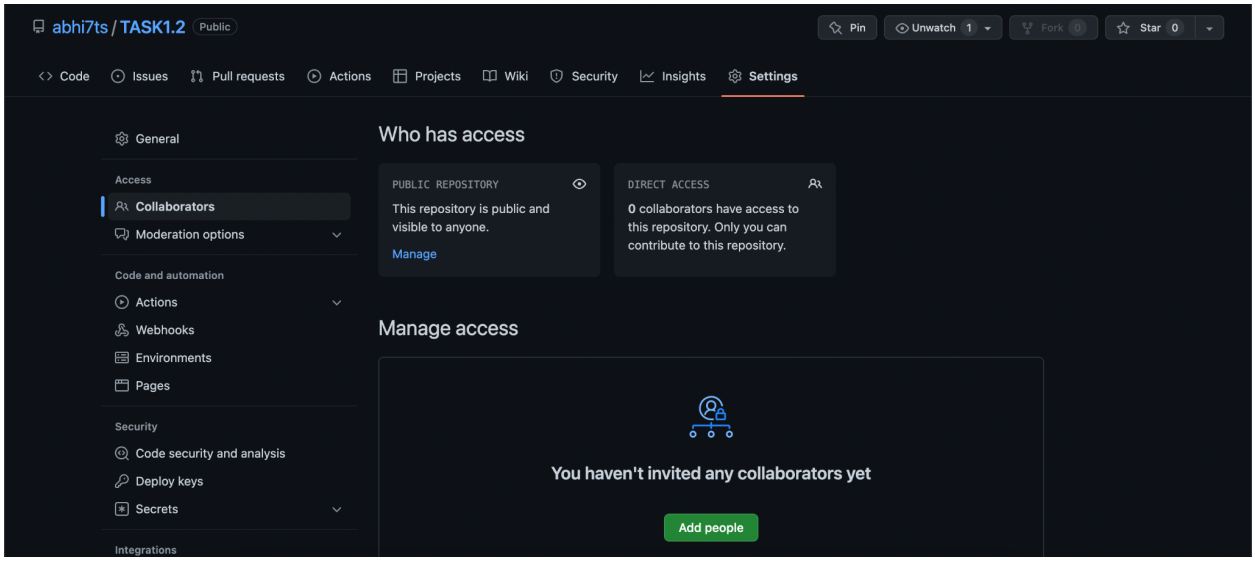
- On your GitHub account, create a new repo.



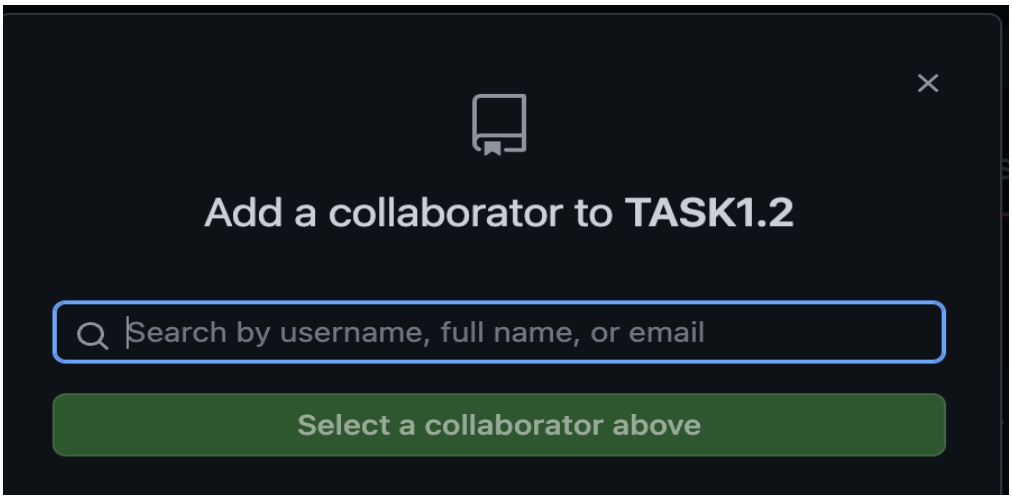
- Open repo and click on settings in the menu bar.



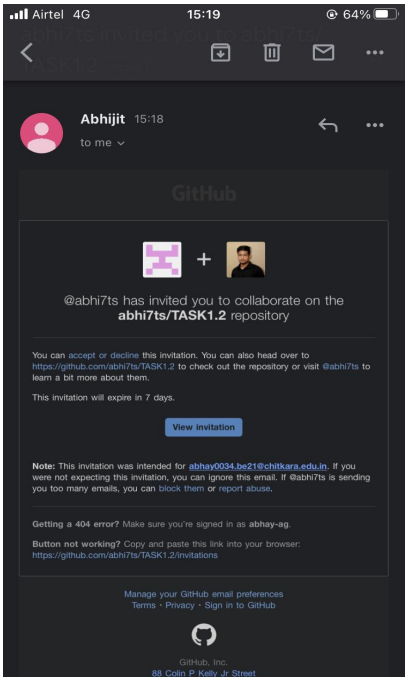
- In the access section of the sidebar, click on the collaborators and a confirm page would appear asking you to enter the password, after entering password it will show who all have the access to this repository.



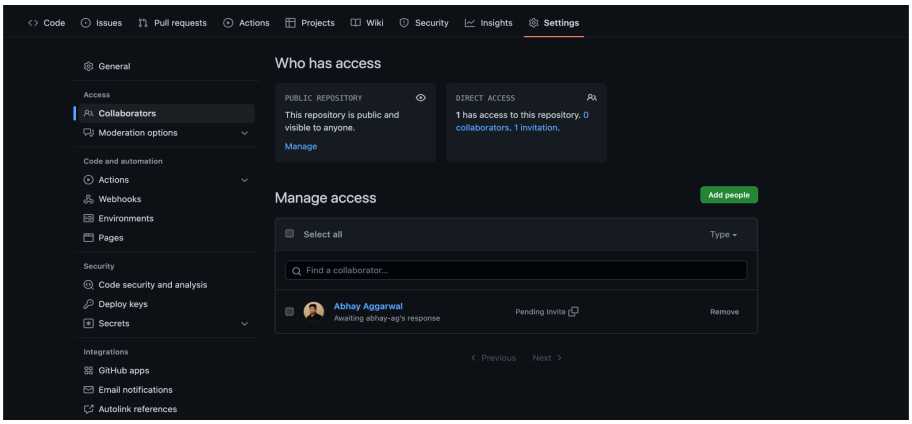
- Click on add people and a box will appear on the screen. Enter or search the username of the collaborators you want to add to the repository.



- An invitation Mail will be sent to the Collaborator which will expire after 7 days if not accepted.



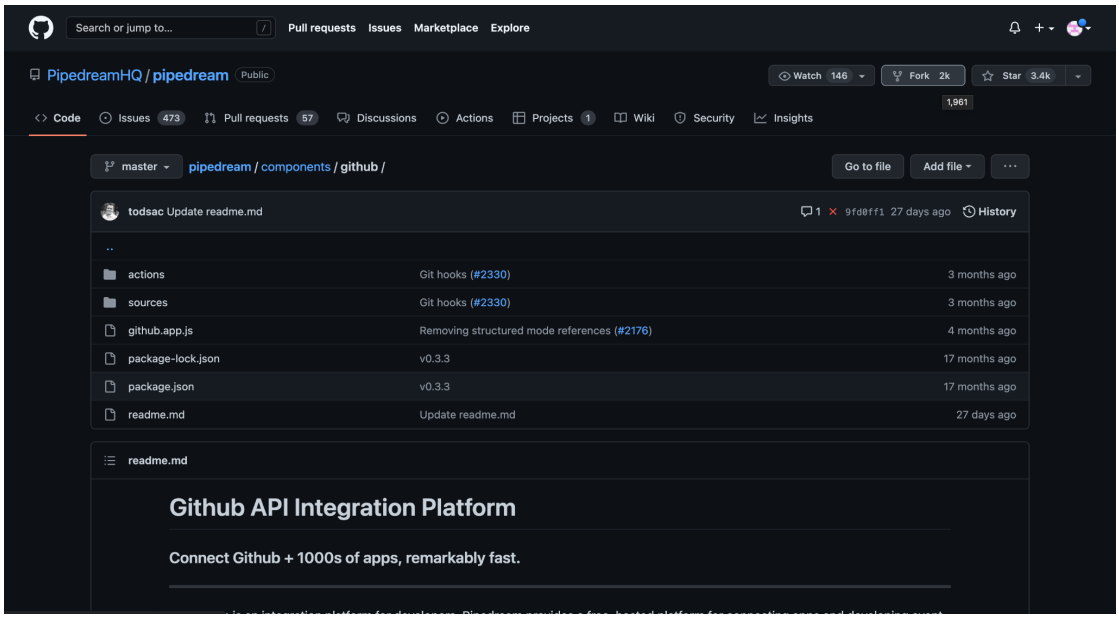
- If the user accepts the invitation, then the collaborators will have access to this repository.

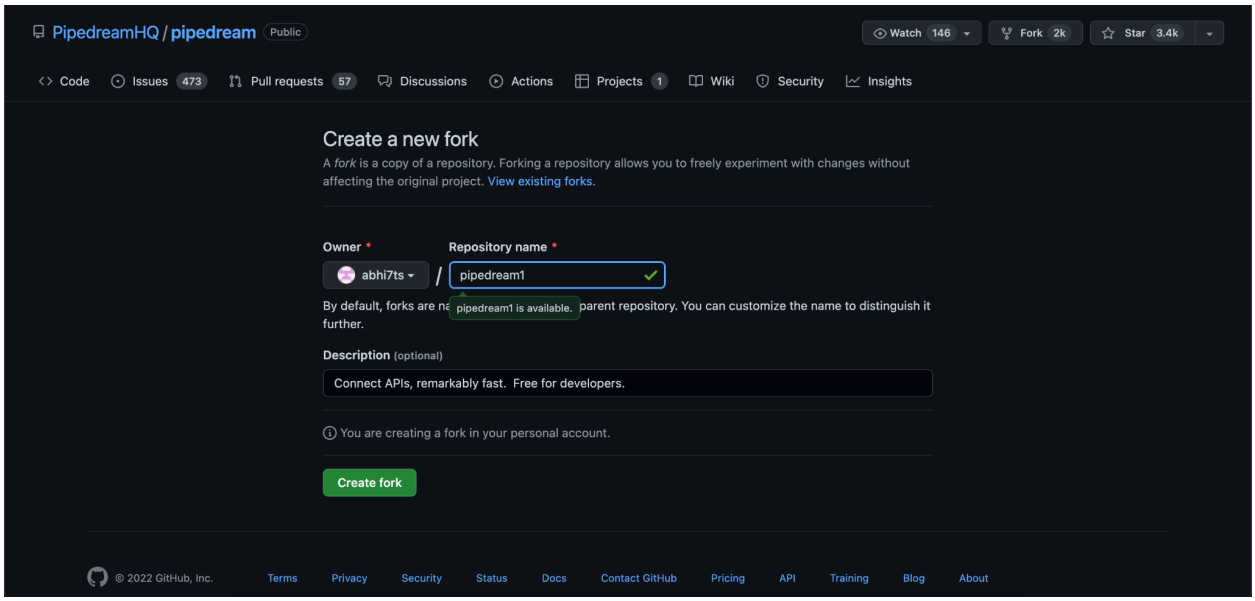


Experiment 2

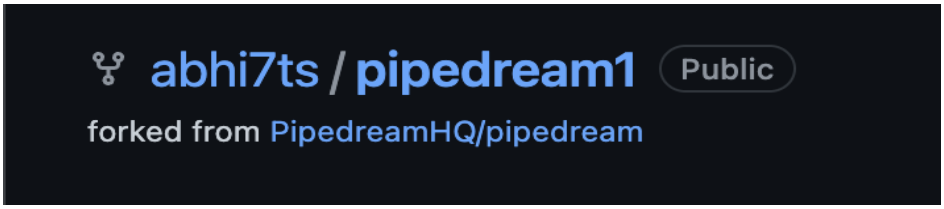
AIM: Fork and Commit.
THEORY: A fork is a copy of a repository that you manage. It allows us to freely experiment with the data. After creating a fork, we can make any desired change like adding collaborators, rename files, generate GitHub pages but all these changes won't be reflected in the original repository. To import the changes into the original repository, the user needs to send a pull request to the maintainer. If the maintainer closes the pull request only then the content can be added to the original repository. Forking is a better method than directly cloning any repository, as in cloning only the default branch is cloned whereas forking creates a clone of the complete repo and also allows us to push the changes to the main repository by using open and close pull requests.
PROCEDURE:

- On GitHub.com, navigate to any repository. Find the fork button on the top right corner. Click on fork and then on the bottom of the page , click on ' Create a Fork'.

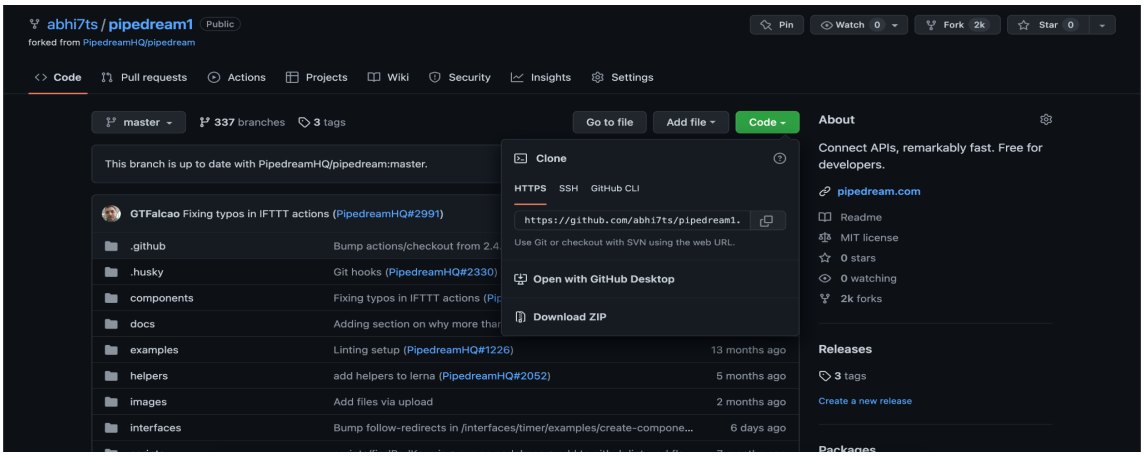




- Now you have the copy of the repository and the name of the repository will be displayed along with your username.



- Further to make changes in the content, you first need to bring the repository on your local machine. We can do so by using the clone command. Click on the green button with the label ‘code’. In the dropdown , the link of the repository will be displayed. Copy that link.



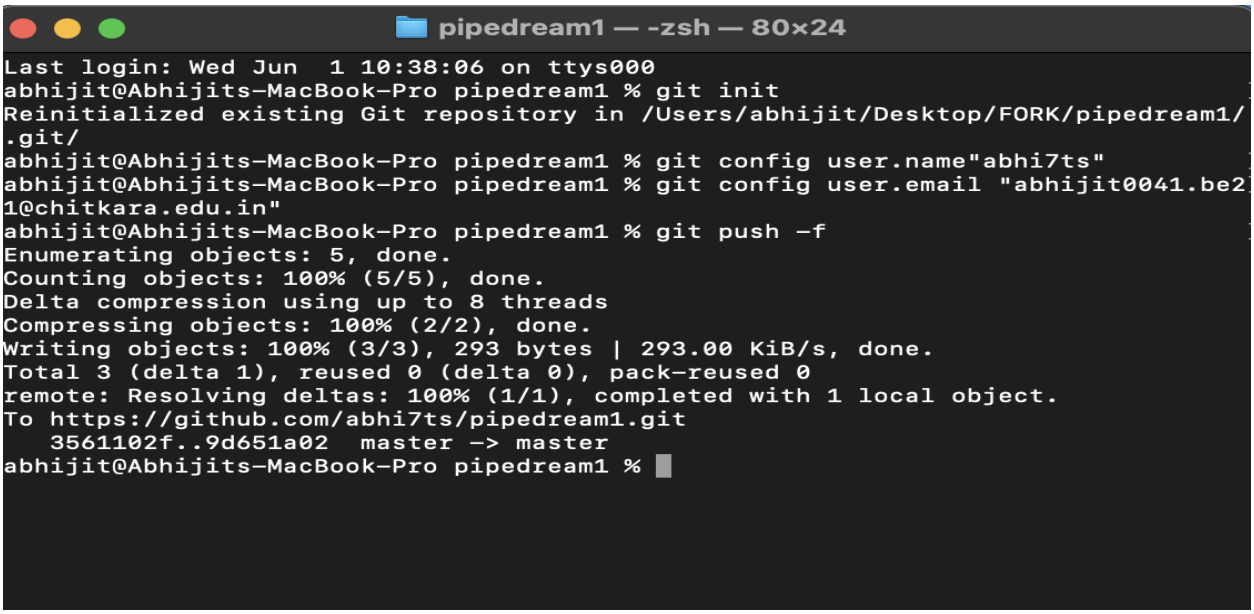
- Create a folder and open the Git Client. Use clone command and paste the URL of the repository.

```
Last login: Sun May 29 22:53:05 on console
abhijit@Abhijits-MacBook-Pro FORK % git init
hint: Using 'master' as the name for the initial branch. This default branch nam
e
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/abhijit/Desktop/FORK/.git/
abhijit@Abhijits-MacBook-Pro FORK % git clone https://github.com/abhi7ts/pipedre
am1.git
Cloning into 'pipedream1'...
remote: Enumerating objects: 39655, done.
remote: Counting objects: 100% (250/250), done.
remote: Compressing objects: 100% (126/126), done.
remote: Total 39655 (delta 107), reused 230 (delta 90), pack-reused 39405
Receiving objects: 100% (39655/39655), 139.13 MiB | 1.58 MiB/s, done.
Resolving deltas: 100% (20464/20464), done.
abhijit@Abhijits-MacBook-Pro FORK %
```

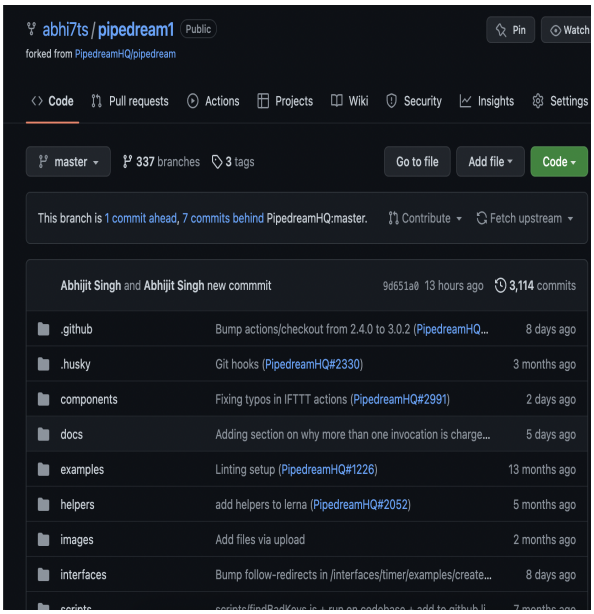
- The repository is now cloned to your local machine and you make any change in the files.
- Stage and commit the changes in the file.

```
abhijit@Abhijits-MacBook-Pro FORK % git commit - "this is the first change"
error: pathspec '-' did not match any file(s) known to git
error: pathspec 'this is the first change' did not match any file(s) known to gi
t
abhijit@Abhijits-MacBook-Pro FORK % git commit -m "this is the first change"
[master (root-commit) d25af46] this is the first change
 Committer: Abhijit Singh <abhijit@Abhijits-MacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 2 files changed, 1 insertion(+)
 create mode 100644 .DS_Store
 create mode 160000 pipedream1
abhijit@Abhijits-MacBook-Pro FORK %
```

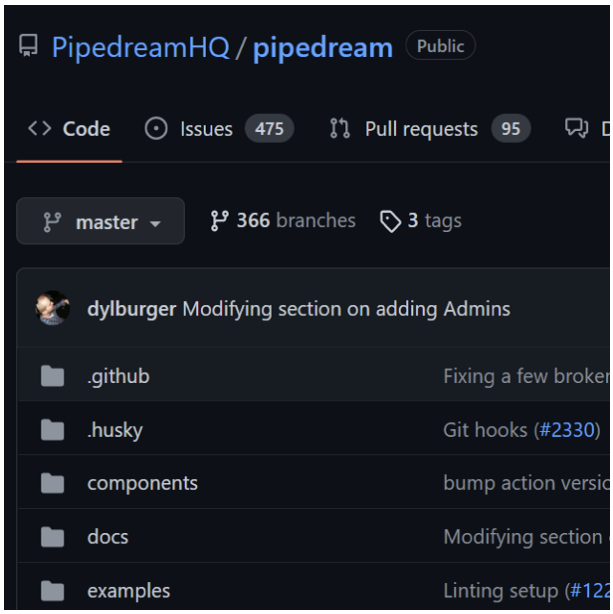
- Now make a remote to create the link between the local repository and the GitHub repository and use the git push command to push the changes on the hub.



- Now observe that the recent commits are displayed only in the forked repository and the original repository remains unaffected.



My repository



Original repository

Experiment 3

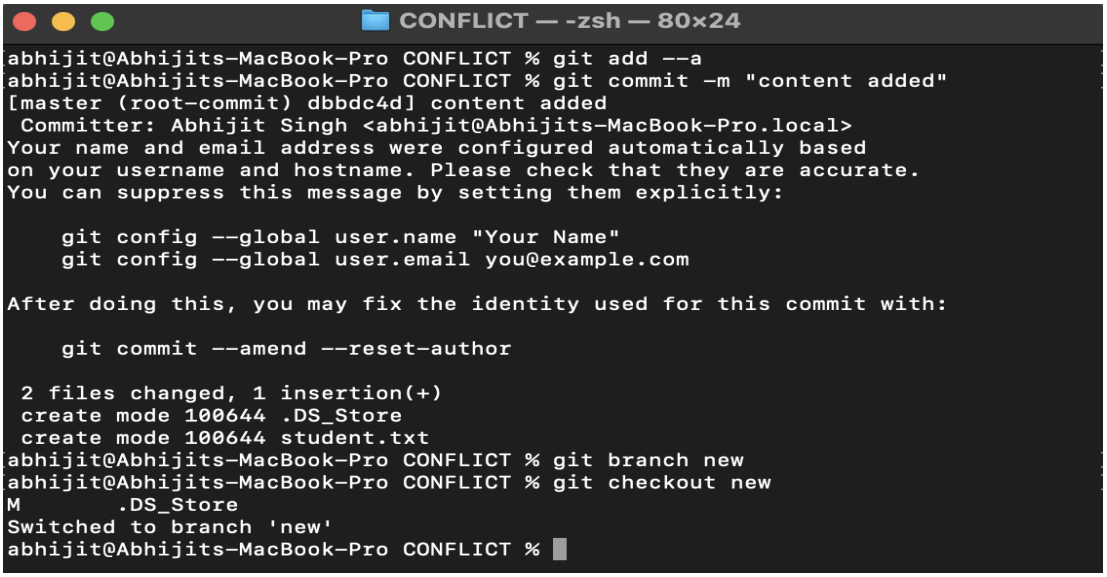
AIM: Merge and Resolve conflicts created due to own activity and collaborators activity.

THEORY: Version control systems are all about managing contributions between multiple distributed authors (usually developers). Sometimes multiple developers may try to edit the same content. If Developer A tries to edit code that Developer B is editing a conflict may occur. If you have a merge conflict on the command line, you cannot push your local changes to GitHub until you resolve the merge conflict locally on your computer. To alleviate the occurrence of conflicts developers will work in separate isolated branches. If a merge conflict still arises between the compare branch and base branch in your pull request, you can view a list of the files with conflicting changes above the Merge pull request button. The Merge pull request button is deactivated until you've resolved all conflicts between the compare branch and base branch.

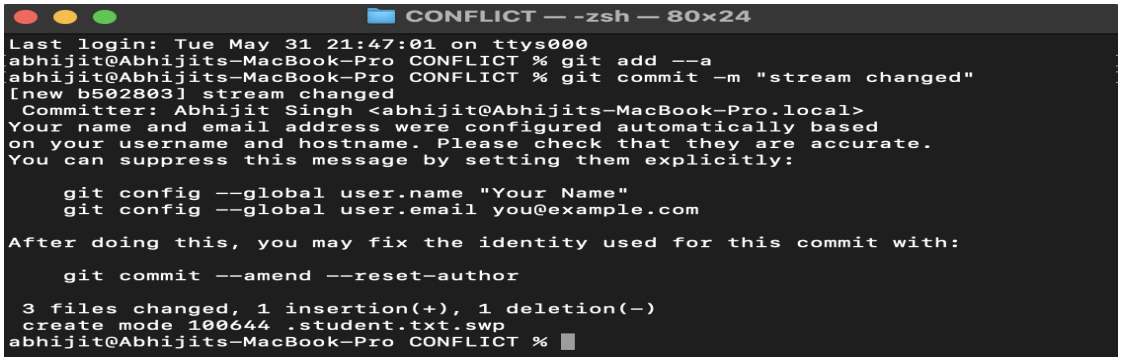
PROCEDURE:

- Make a folder and open terminal at the folder. Initialize a git repo and create a file in it and add some content in the file.

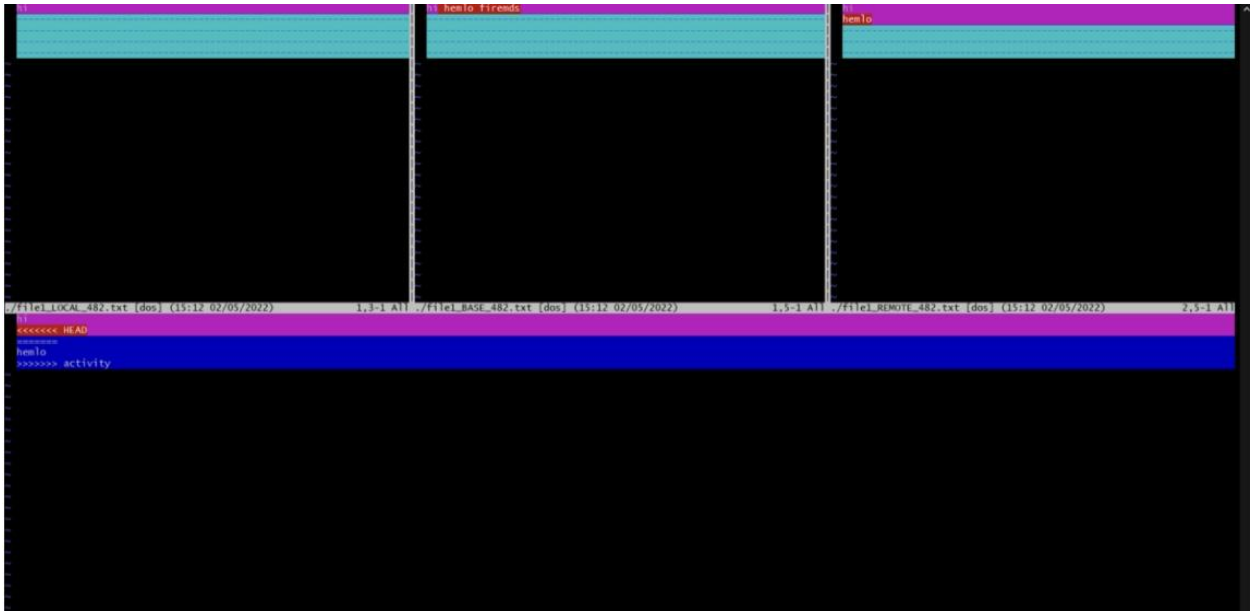
- Stage and commit the modifications in the file. Create a new branch. Checkout to the new branch and make some changes in the same file.



- Open the same file and make some changes. After making the change we need to stage it and commit.



- Checkout to the master branch. Again, open the same file and again make some changes. Stage and commit the changes.
- Now merge the two branches and a window will appear. Click ‘l’ keep the content you want to keep and delete the rest then enter the “esc” button and write the command “:wq”.



Experiment 4

AIM: Reset and Revert.
THEORY: A reset is an operation that takes a specific commit and resets the "three trees" to match the state of the repository at that specific commit. A reset can be invoked in three different modes which correspond to the three trees. In reset, the rest of the commits wash out after the mentioned commit. This is a limitation of the reset command that we cannot have any random access. A revert is an operation that takes a specific commit and creates a new commit which inverses the specific commit. git revert can only be run at a commit level scope and has no file level functionality. These two features justify the Version- controlled feature of the git as we can rollback to any version at any time.
PROCEDURE: RESET:

- Create a few files, stage them and commit .

```
abhijit@Abhijits-MacBook-Pro RR % git log --oneline
844e69e (HEAD -> master) ordered added
0672bcd homepage created
66ec0a5 contactus page added
4adafcc feedback form created
f7e542e menssection added
56cef07 kidssection added
abhijit@Abhijits-MacBook-Pro RR %
```

- Pick any commit where you want the repository to rollback. Copy its checksum and paste it in the **git reset checksum** command .

```
abhijit@Abhijits-MacBook-Pro RR % git log --oneline
0672bcd (HEAD -> master) homepage created
66ec0a5 contactus page added
4adafcc feedback form created
f7e542e menssection added
56cef07 kidssection added
abhijit@Abhijits-MacBook-Pro RR %
```

The head is now pointing to the commit whose checksum we have provided that means the commits that followed vanished.

- If you want to undo this change, you copy the checksum of the commit you want back and run the same command again.

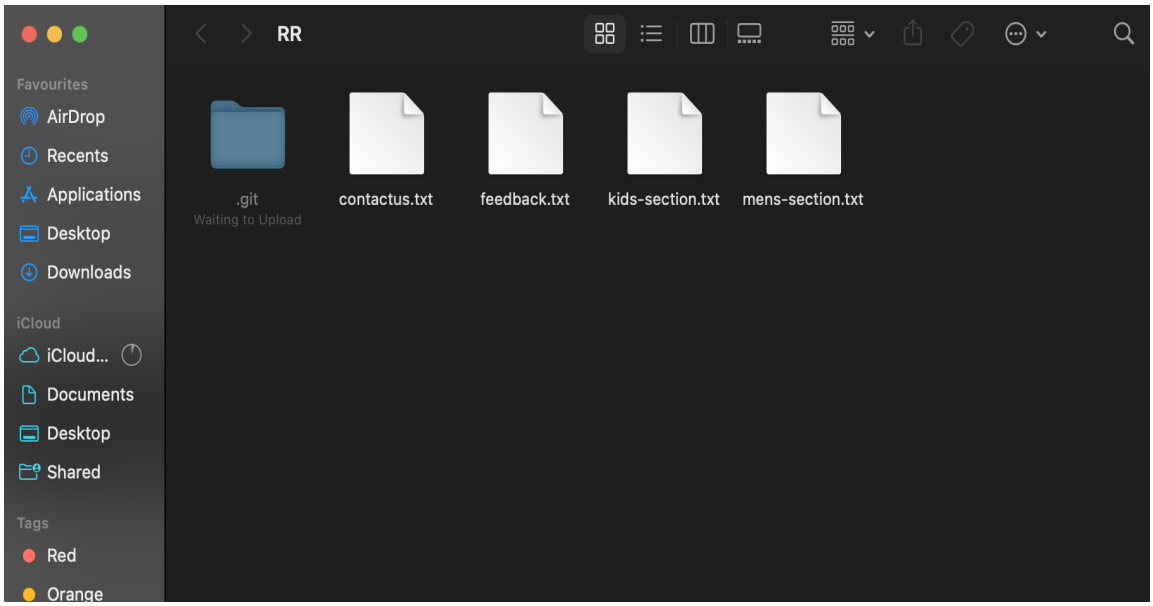
```
abhijit@Abhijits-MacBook-Pro RR % git log --oneline
844e69e (HEAD -> master) ordered added
0672bcd homepage created
66ec0a5 contactus page added
4adafcc feedback form created
f7e542e menssection added
56cef07 kidssection added
```

TYPES OF RESET:

1. **git reset checksum —hard:**
   This command removes the file from the working space and staging area.

```
abhijit@Abhijits-MacBook-Pro RR % git reset 66ec0a5 --hard
HEAD is now at 66ec0a5 contactus page added
abhijit@Abhijits-MacBook-Pro RR % git reset 844e69e
Unstaged changes after reset:
D       homepage.txt
D       ordered.txt
abhijit@Abhijits-MacBook-Pro RR %
```

   Also the changes or the files that were associated to those commits are also removed from the folder.



To remove ‘n’ number of commits from the last, from working space.

```
abhijit@Abhijits-MacBook-Pro RR % git reset --hard Head~2
HEAD is now at 66ec0a5 contactus page added
abhijit@Abhijits-MacBook-Pro RR % git log --oneline
66ec0a5 (HEAD -> master) contactus page added
4adafcc feedback form created
f7e542e menssection added
56cef07 kidssection added
abhijit@Abhijits-MacBook-Pro RR %
```

2. **git reset checksum —mixed:**
   This command unstages the commits made after the targeted checksum. The commits go in the working space.

```
abhijit@Abhijits-MacBook-Pro RR % git reset 4adafcc --mixed
abhijit@Abhijits-MacBook-Pro RR % git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .DS_Store
        contactus.txt

nothing added to commit but untracked files present (use "git add" to track)
abhijit@Abhijits-MacBook-Pro RR %
```

   To unstage ‘n’ number from last , we can use the command git reset – mixed Head~n.

```
abhijit@Abhijits-MacBook-Pro RR % git reset --mixed Head~2
abhijit@Abhijits-MacBook-Pro RR % git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .DS_Store
        contactus.txt
        feedback.txt
        mens-section.txt

nothing added to commit but untracked files present (use "git add" to track)
abhijit@Abhijits-MacBook-Pro RR %
```

3. **git reset checksum –soft:**
   This command brings the commands to the staging area.

```
abhijit@Abhijits-MacBook-Pro RR % git reset 56cef07 --soft
abhijit@Abhijits-MacBook-Pro RR % git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .DS_Store
        new file:   contactus.txt
        new file:   feedback.txt
        new file:   mens-section.txt
```

PROCEDURE: REVERT:

- Pick the change where you want the project to revert back. Copy its checksum and paste it in the revert command.

```
abhijit@Abhijits-MacBook-Pro R % git log --oneline
81c070c (HEAD -> master) words added
2ce7b17 few lines added
c27e36a added more content
3b8f65a added content
4306c6e homepage created
abhijit@Abhijits-MacBook-Pro R % git revert 2ce7b17
```

- A window will appear. Press ‘i’ and write the statement you want to be displayed for reverting the change. The change associated with the reverted commit has disappeared. after completing press ‘esc’ and write :wq in the terminal.

```
R — vi ◂ git revert 2ce7b17 — 80×24
Revert "i want this change"

This reverts commit 2ce7b1735de7448e2577d3c76f93e74fd4f4b5c7.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#       deleted:    few.txt
#
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

- Check the git log and you will find another commit is added without affecting the rest commits. The change associated to the reverted commit has disappeared.

```
abhijit@Abhijits-MacBook-Pro R % git log --oneline
de1630c (HEAD -> master) Revert "i want this change"
81c070c words added
2ce7b17 few lines added
c27e36a added more content
3b8f65a added content
4306c6e homepage created
```