**List of Programs**

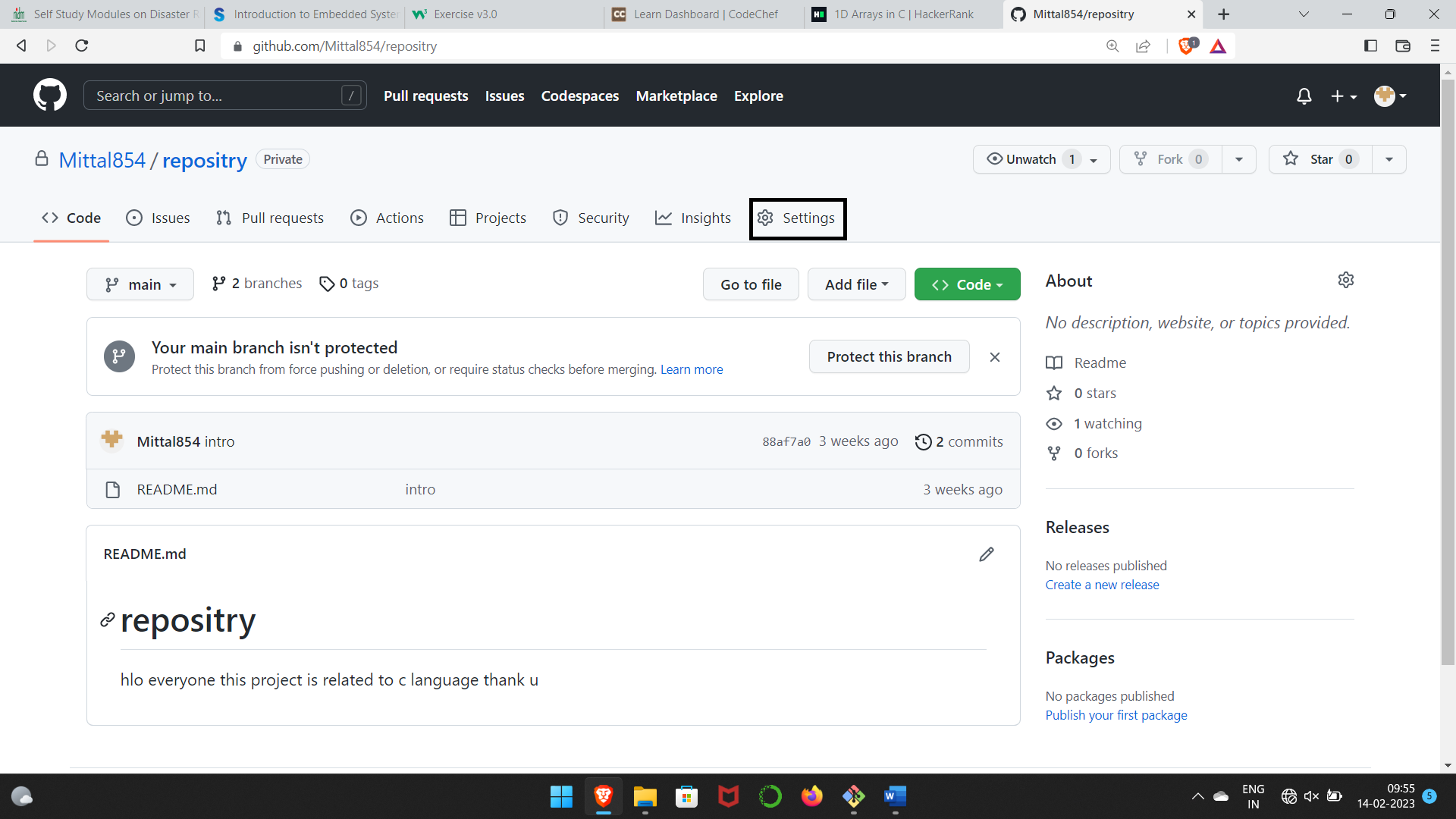
|  |  |  |
| --- | --- | --- |
| **S. No** | **Program Title** | **Page No.** |
| 1 | Setting up of Git Client | 3-4 |
| 2 | Setting up GitHub Account | 5-6 |
| 3 | Generate logs | 7 |
| 4 | Create and visualize branches | 8-10 |
| 5 | Git life cycle description | 11-12 |
| 6 | Add collaborators on GitHub Repo | 13-14 |
| 7 | Fork & commit | 15-17 |
| 8 | Merge and Resolve conflicts created due to own activity and collaborators activity. | 18-20 |
| 9 | Reset and revert | 21-24 |

Experiment 6. Add collaborators to GitHub Repo

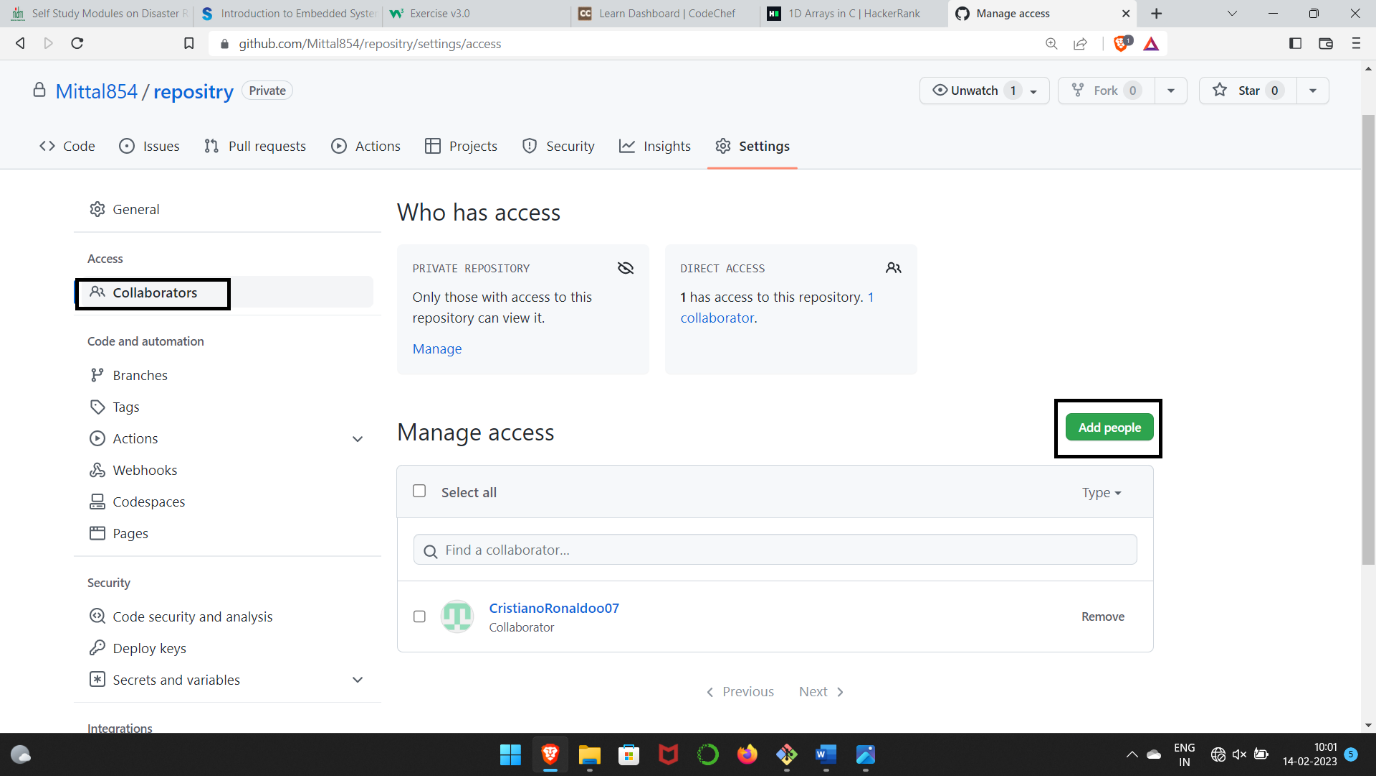
**Step 1.** Open the GitHub website: [***https://github.com/***](https://github.com/) and log in to your account.

**Step 2.** Create a new repository.

**Step 3.** After creating a repository, go to settings.

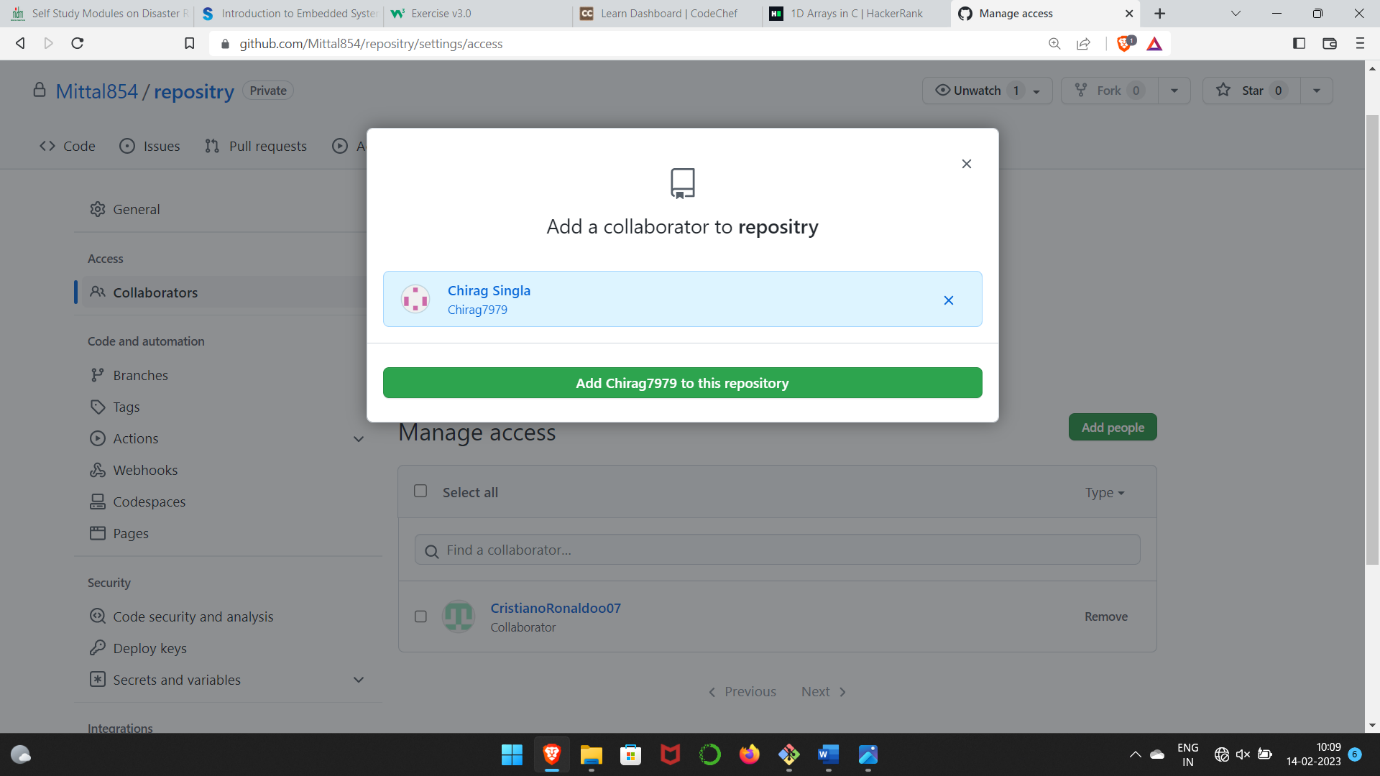


**Step 4.** Go to collaborators & click on add people.



13

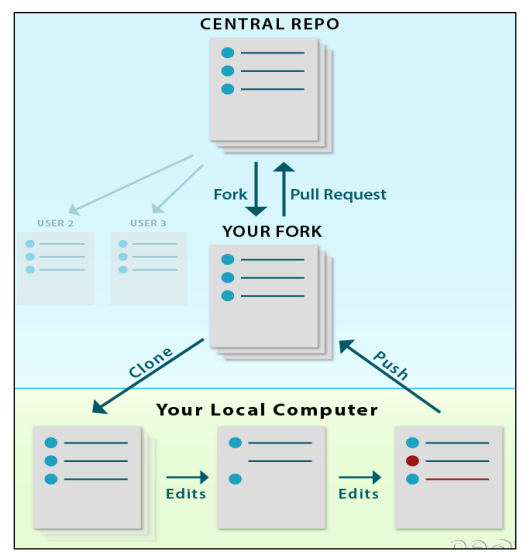
**Step 5.** Search the collaborators name and add him to the repository.



14

Experiment 7. Fork & Commit

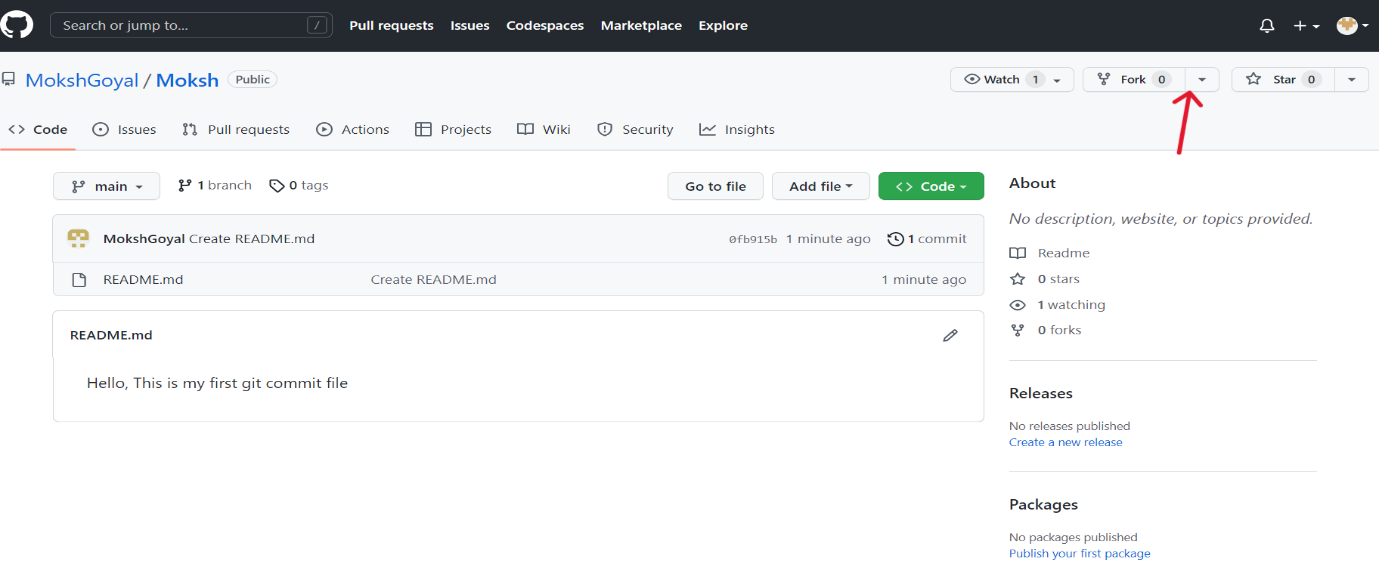
A fork is a copy of a repository that you manage. It allows us to freely experiment with the data. After creating a fork, we can make any desired change like adding collaborators, rename files, generate GitHub pages but all these changes won’t be reflected in the original repository.



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.

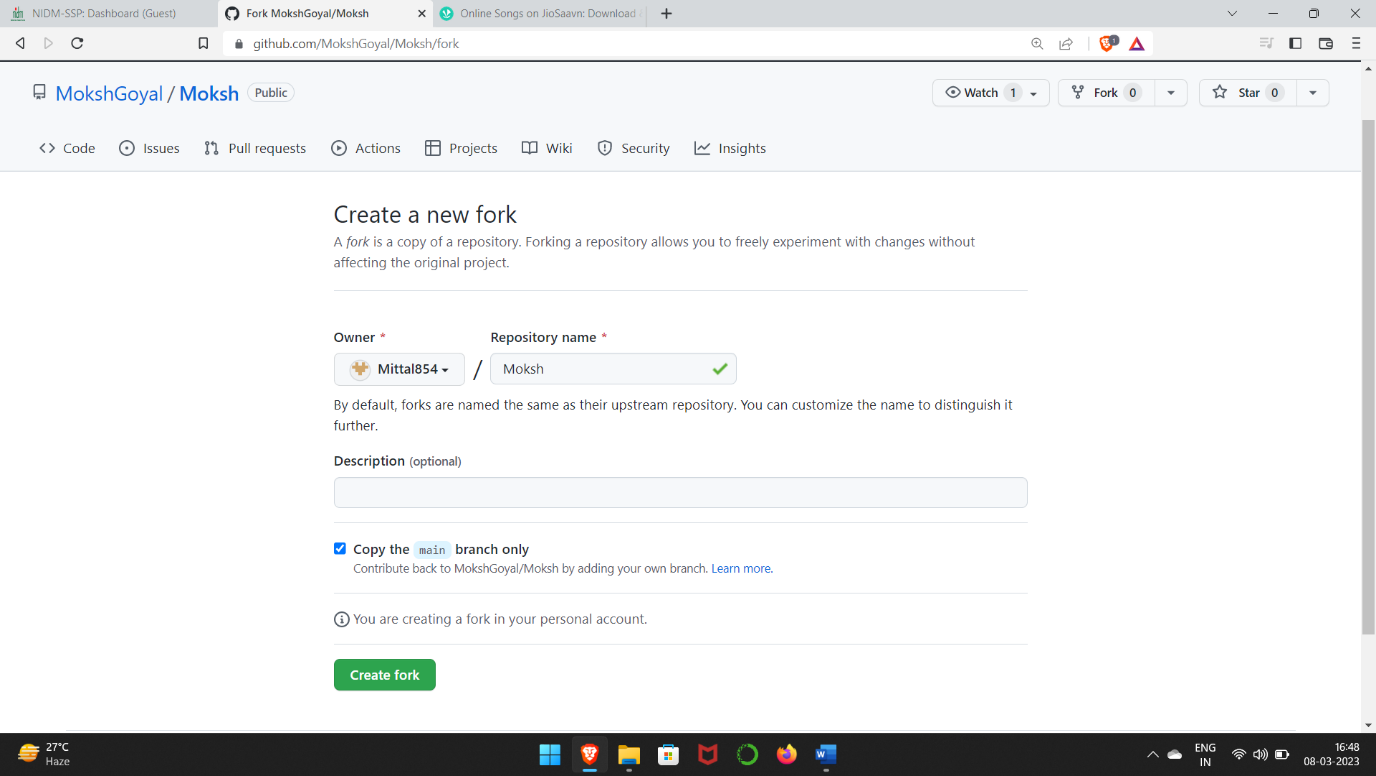
Forking is a better method than directly cloning any repository, as in cloning only the default branch is cloned whereas forking creates a clone of the complete repo and also allows us to push the changes to the main repository by using open and close pull request

**Step 1.** Open the repository which you want to fork & click on fork option.



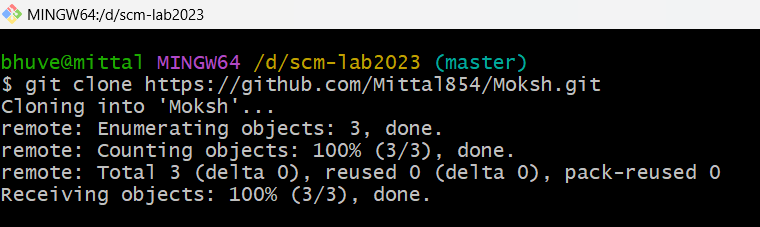
15

**Step 2.** Click on create fork.



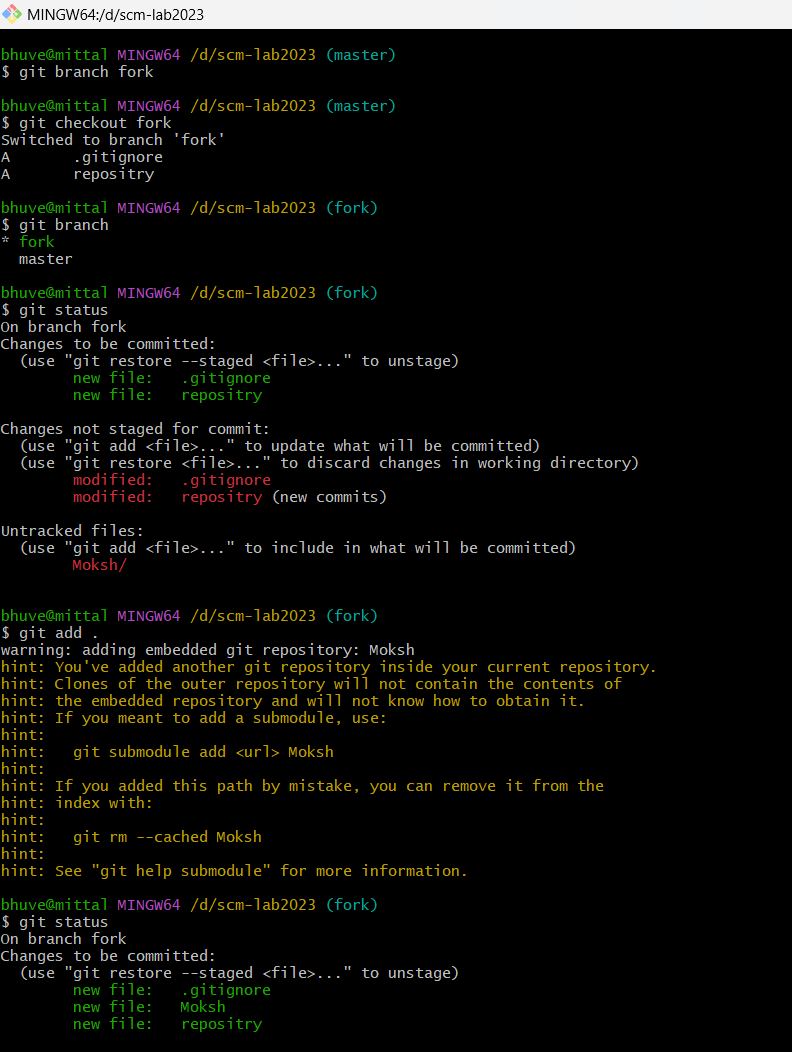
**Step 3.** A copy of the repo which is forked from other user is formed. We can now make modification without changing main source code.

**Step 4.** Now use the command **git clone <URL>** to fetch the remote repo or clone the repo.



**Step 5.**  Now we can open the file, make changes & commit.

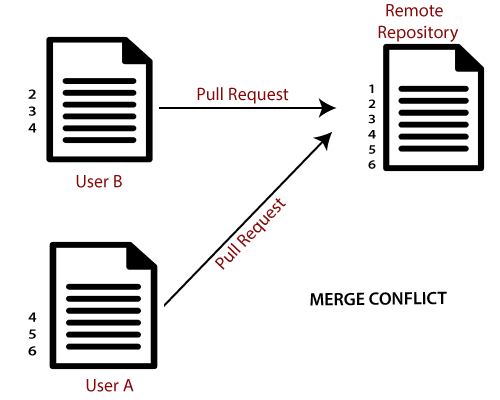
16



17

Experiment 8. Merge and Resolve conflicts created due to own activity and collaborators activity.

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. reference for picture:

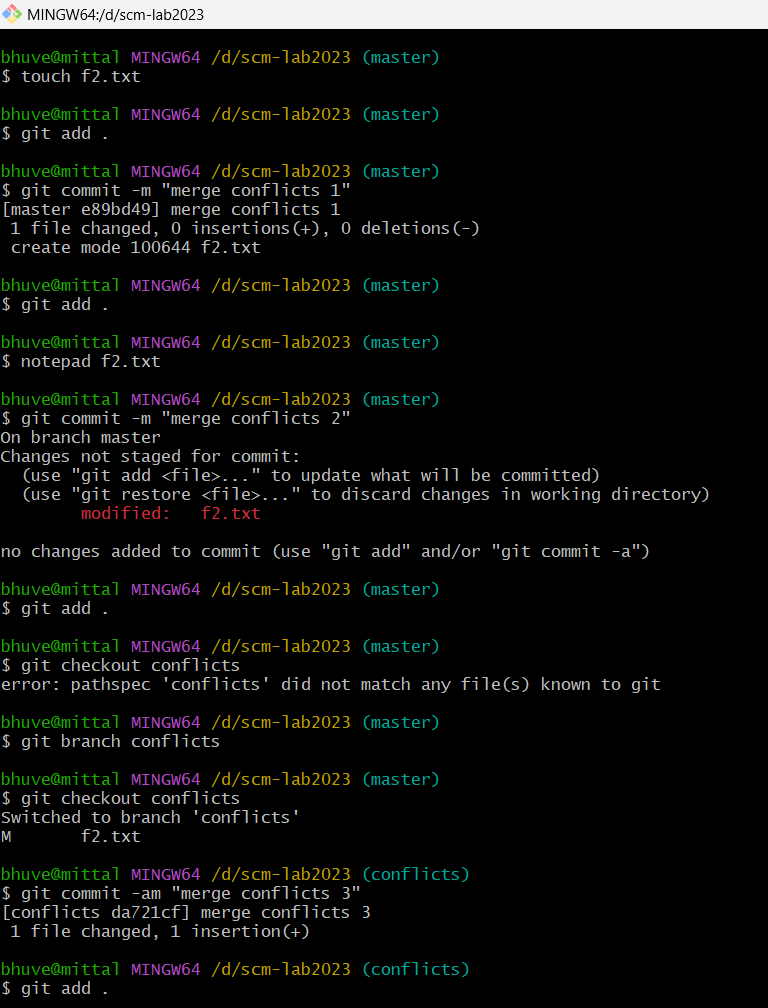


If you have a merge conflict on the command line, you cannot push your local changes to GitHub until you resolve the merge conflict locally on your computer.

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.

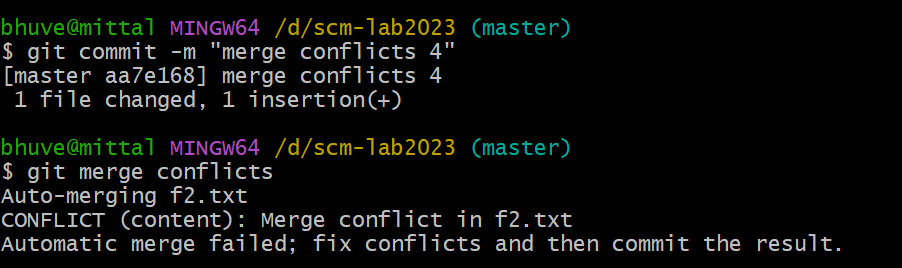
18

**Step 1.** Make changes in the master branch & commit them. Now checkout to another branch, make changes & commit them also.

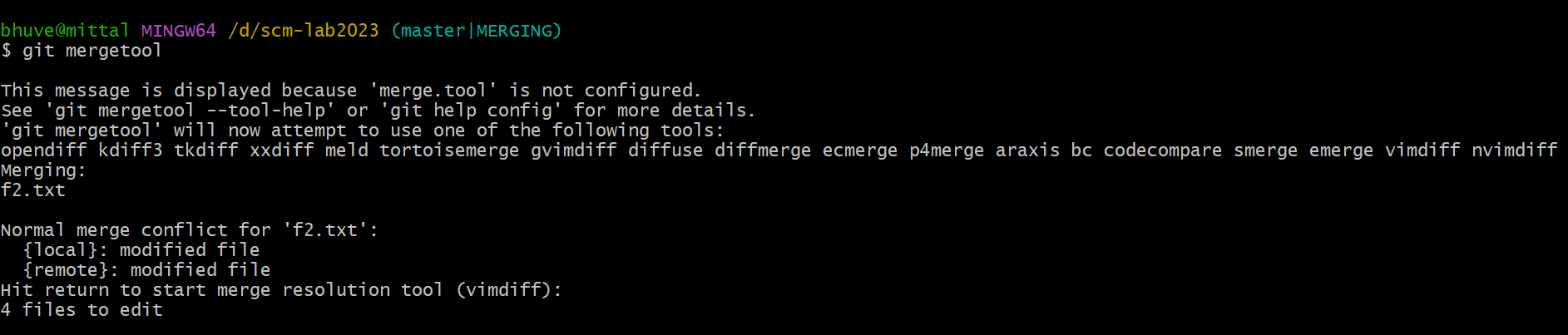


19

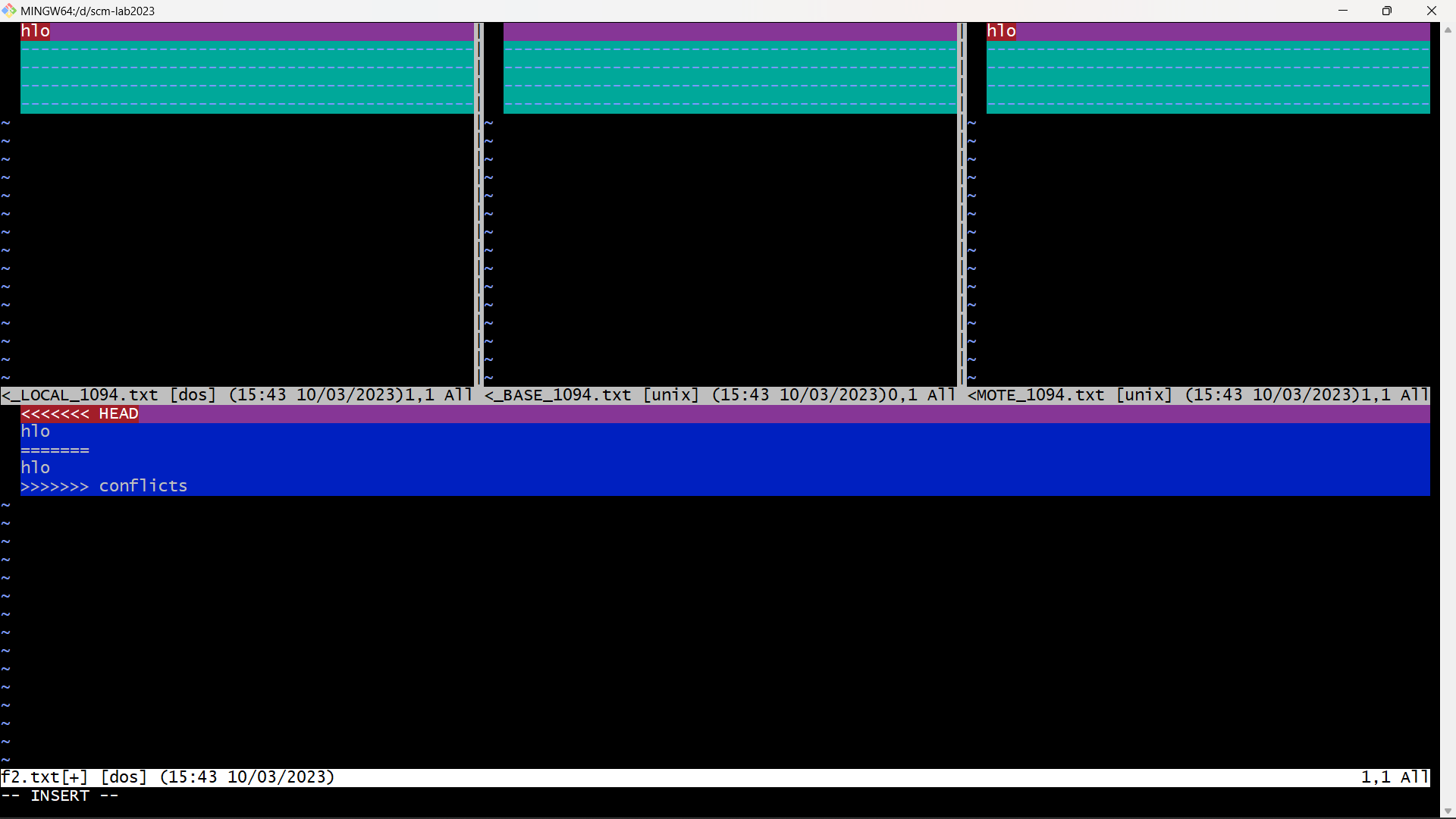
**Step 2.**  Now merge both the branches. It will give Conflicts error.



**Step 3.** Use **git mergetool** command to resolve the conflict. It runs the merge conflict resolution tools to resolve merge conflicts.

****

**Step 4.** To insert, press **I**. Then type **:wq**. The merge conflict is solved and secondary branch is merged into the master branch.



20

Experiment 9. Reset and Revert

A reset is an operation that takes a specified commit and resets the "three trees" to match the state of the repository at that specified commit. A reset can be invoked in three different modes which correspond to the three trees. In reset, rest of the commits wash out after the mentioned commit. This is a limitation of reset command that we cannot have any random access.

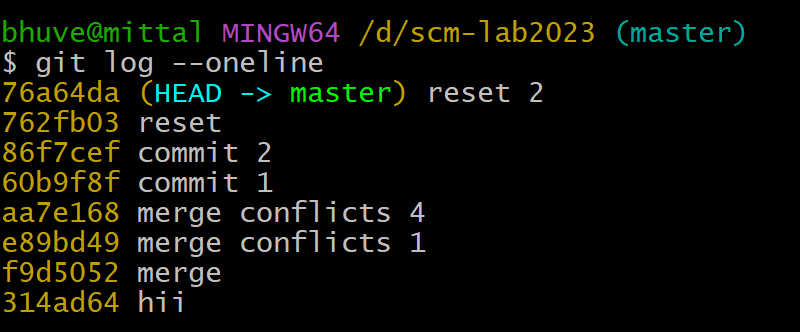
A revert is an operation that takes a specified commit and creates a new commit which inverses the specified commit. git revert can only be run at a commit level scope and has no file level functionality.

These two features justify the Version- controlled feature of the git as we can rollback to any version at any time.

**Reset: -**

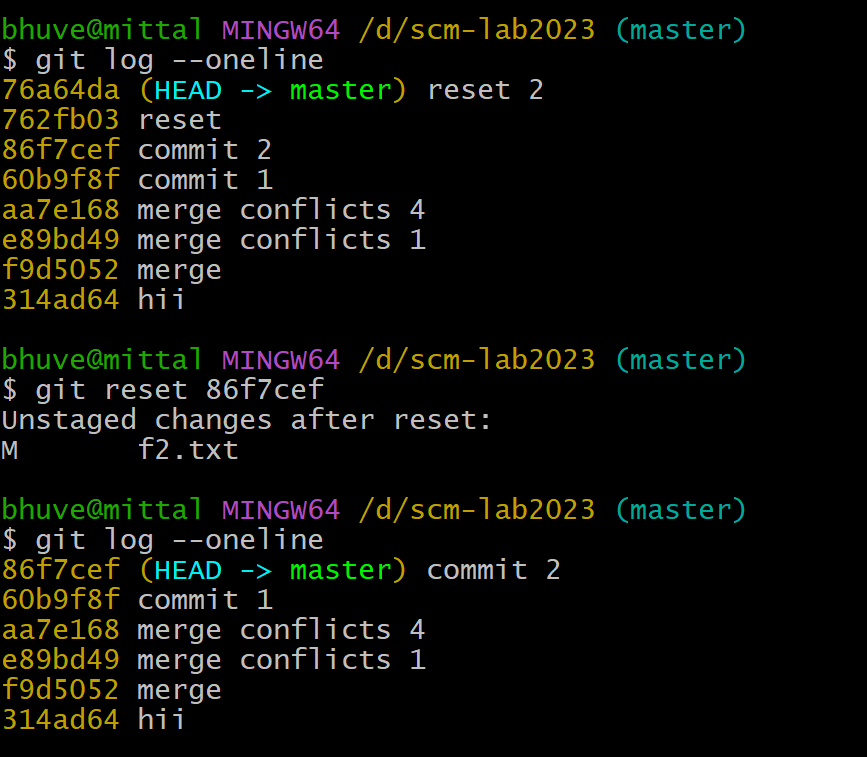
**Step 1.** Prepare a log of multiple commits to use reset command.

**Step 2.** Check git log.



**Step 3.** Pick any commit which we want the repository to rollback. Paste the checksum of the commit in the **git reset** command.

21



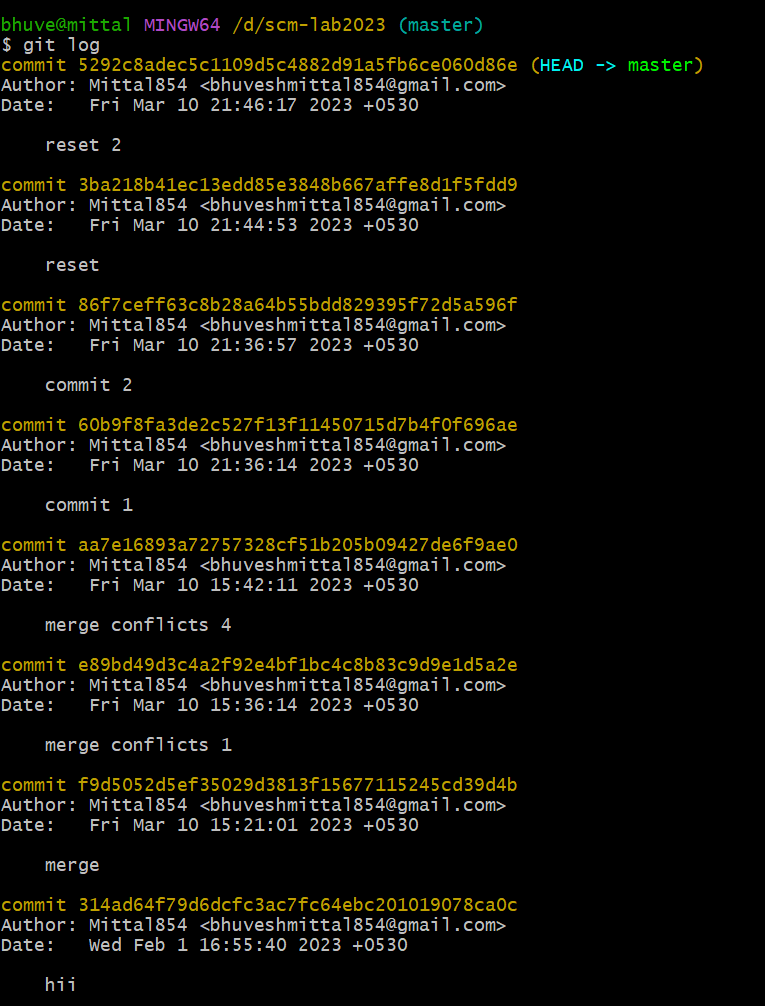
The head is now pointing the commit whose checksum we have provided that means the commit that followed vanished.

**Revert: -**

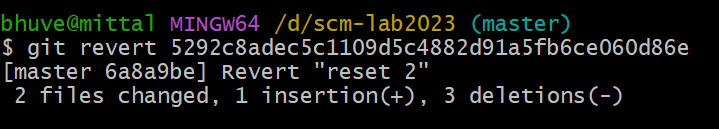
**Step 1.** Prepare a log of multiple commits to use reset command.

**Step 2.** Check git log.

22



**Step 3.** Pick up the commit which we want to revert. Paste the checksum in the **git revert** command.



23

**Step 4.** A window will appear. Enter the statement which we want to be displayed after reverting the commit.

**Step 5.** Check the git log. We will find that another commit is added without affecting the rest commits.



The change associated to the reverted commit has disappeared.

24