# Experiment No. : 3

**Title:** Implementation of Commit, Modification on Branch Master, Forking and GitHub Repository Link.

# Objectives:

To implement Commit, Modification on Branch Master, Forking and GitHub Repository Link .

# Theory:

# In Git, a branch is a new/separate version of the main repository.

Let's say you have a large project, and you need to update the design on it.

How would that work with Git:

* With a new branch called new-design, edit the code directly without impacting the main branch
* EMERGENCY! There is an unrelated error somewhere else in the project that needs to be fixed ASAP!
* Create a new branch from the main project called small-error-fix
* Fix the unrelated error and merge the small-error-fix branch with the main branch
* You go back to the new-design branch, and finish the work there
* Merge the new-design branch with main (getting alerted to the small error fix that you were missing)

Branches allow you to work on different parts of a project without impacting the main branch. When the work is complete, a branch can be merged with the main project. You can even switch between branches and work on different projects without them interfering with each other

Git GitHub Fork

At the heart of Git is collaboration. However, Git does not allow you to add code to someone else's repository without access rights.

A fork is a copy of a repository. This is useful when you want to contribute to someone else's project or start your own project based on theirs.

fork is not a command in Git, but something offered in GitHub and other repository hosts.

Implementation steps:

**Clone remote Repo**

git clone “URL-of-Remote-Repository” // use current directory

git clone “URL-of-Remote-Repository” desktop/gitrepo1 // use specified directory

**View all branches**

git branch

git branch –a // display all branches remote and local

**Create new branch**

git branch mergebranch

git branch rebasebranch

**Go to New branch (to make changes)**

git checkout mergebranch

\*\*\*\*\* add one file and commit \*\*\*\*\*\*

**Push branch to remote repo**

git push origin mergebranch

git push --all origin // Push all Branches to remote

git branch –a // Display all the branches

**Merge Branches** (into master )

git checkout master // Go to master branch

git pull origin master // Pull changes if any

git branch --merged // Display Merged branches if any

git merge mergebranch // Merge mergebranch into master

**Rebase Branches (into master**)

git checkout master

\*\*\*\*\* add one file and commit \*\*\*\*\*\*

git rebase rebasebranch

**Deleting a Branch**

git branch --merged // Display Merged branches if any

git branch –d mergebranch // Delete Branch on local repo

git branch –a // Display Branches

git push origin -–delete mergebranch // Delete Branch on remote repo

**Undo the changes made** **(in staging area but not commited)**

git checkout filename.txt

**Edit the commit message (changes the hash – recommended if changes made are not**

**pushed to central repo)**

git commit --stat // See the Stats

git commit --amend -m “Updated Message” // Change only the commit message

**\*\* Make changes in a file and add one file \*\***

git commit –-amend // to add files modified or newly added to previous commit

**Move commit to a different branch**

git checkout newbranch // Go to that branch

git cherry-pick commit-hash

\*\* delete commit from original branch \*\*

git reset --soft commit-hash //check the log to see changes (Hash should be

one commit old)

git reset commit-hash // mixed (moves changes to working directory)

git reset –-hard commit-hash // (Hash should be one commit old)

git clean –df // gets rid of untracked directories and files

**Get back files before being collected by the garbage collector**

git reflog

git checkout commit-hash

git log

git branch backup

git checkout master

git branch

git checkout backup

**Revert commit (undo changes that others have pulled)**

git revert commit-hash // save and exit the message

git diff commit-hash revert-hash

**Stash Command (when not ready to commit but save changes)**

git stash save “Stash-message”

git stash list

git stash apply stash-code // Restore changes but keep stash (applicable between

branches)

git checkout -- . // Go back to state before apply

git stash pop // Restore changes most recent and remove stash (applicable between

branches)

git stash drop stash-code // Delete specific stash

git stash clear // Clear all stashe