**GIT KATAS ASSIGNMENT**

1. Git add and Git commit

Git add command is used to stage the untracked files

Git commit command is used to commit the added changes with a commit message.

Git status provide the status information like on which branch and details about files which are tracked or untracked.

1. Git basic Staging

Git diff tells shows changes between the working tree and the index or a tree, changes between the index and a tree, changes between two trees, changes resulting from a merge, changes between two blob objects, or changes between two files on disk.

Git diff –staged will only show changes to files in the staged area.

Git restore –staged file.txt unstages the changes from file.txt file.

Git restore file.txt restores file.txt specified paths in the working tree with some contents from a restore source. If a path is tracked but does not exist in the restore source, it will be removed to match the source.

1. Git basic branching

Git branch tells us the different branches present and Git switch is used to switch from one branch to another branch.

Git log –oneline –graph –all tells us the graphical representation of various commits and its linkage.

Git branch mybranch creates a new branch named mybranch and we can checkout to that branch using Git checkout mybranch command.

Git diff mybranch master tells us the differences between two branches.

1. Git Fast Forward Merge

feature/uppercase is the correct naming convention for a branch in a git repo.

We can merge two branches using command git merge <branch> in a branch where we want to merge with the other branch.

We can delete the branch using command Git branch -d <branch>

1. Three way Merge

Example we create a file called greeting.txt which contains greeting message. Add and commit this and this process is done in greeting branch. Get back to Master branch and add a READ.me file and add and commit that file. Now, merge greeting with master and when we enter command git log --oneline --graph –all we can see extra merge commit created with message “merge branch ‘greeting’”.

6&7. Merge Conflict

A merge conflict is an event that takes place when Git is unable to automatically resolve differences in code between two commits. Git can merge the changes automatically only if the commits are on different lines or branches.

It happens when the 2 or more developers pull and edit the changes by pushing them back to master. We can resolve by following git status instructions in master by editing using merge tools.

Ex : git mergetool --tool=vimdiff

8. Branch re-base

Rebasing in Git is a process of integrating a series of commits on top of another base tip. It takes all the commits of a branch and appends them to commits of a new branch.

Command for rebasing to master is git rebase master.

9. Basic revert

We use git revert on the newest commit, to remove the changes the last commit added.

We can see those previous commited changes using git show command.

10. Git reset

git log --oneline : This command enables gitbash to show the commits made in single line format

git reset --soft HEAD~1 : It moves HEAD by 1 commit without changing the state of our files

git reset –hard HEAD~1 : It will remove completely the last 1 commit and destroy any changes and removes from local directory also.

git reset –mixed HEAD~1 : It keeps the last commit same but unstages the changes.

git revert HEAD~1 : It undoes the last commit changes to the repository.

11. Basic Cleaning

git clean -n : Removes all untracked files and directories

git clean -n -f :Forced to removes all untracked files

git clean -f -d :Removes untracked directories

12. Amending Commits

git commit --amend : This command enables user to change the name of commit by opening it in a vim editor.

13. Reorder the history

git rebase -i HEAD~3 : It is a sub-branch of rebase command , it opens in a vim editor whwre user can reorder the commits made in last 3(as specified in HEAD~3) save the changes and quit by :wq.

Hit git status and then git rebase –continue to proceed further with the changes made.

git log --oneline --graph --all : shows all branches in graphical format

14. squashing

Squashing is also one of the options provided by rebase command

git rebase -i HEAD~3 : opens last three commits in vim editor pick of 2nd and 3rd can be changed to squash so that 2nd and 3rd commits can be made into single commit with the 1st commit message

and hit git status and follow then continue with git add and git commit then git rebase --continue.

Also fixup can be used for this purpose directly to squash the commits.

15.advanced-rebase-interactive

Opens editor with list of commits to be changed.The list accepts commands allowing the user to edit the list before initiating the rebase action.

16. Basic Stashing

git stash : it temporarily stashes changes made to working directory and can be reapplied later.

17.ignore

.gitignore file is a .txt file in which the files which are to be untracked are stored and saved.if any particular file which is to be tracked from gitignore file , it is to be mentioned particularly in .gitignore file(!file.txt).

18. Git submodules

We can use git submodule add command for adding submodules into modules. We can initialize and update from remote repositories using git submodule init and update commands.

19. Git Tags

It is used for versioning the commits, developers easily identify the commits by their tags.