Install GIT & make sure it is added into PATH.

Section 0 -Use GIT as local VCS. Steps to follow:

1. Create a directory ‘project\_dir’ & cd to ‘project\_dir’.

mkdir project\_dir >> cd project\_dir

1. Initialize git version database. (git init)

C:/Users/PSING229/project\_dir/.git/

1. Create a new file index.html.

touch Index.html

1. Check the git status. You should find index.html as an untracked file.

git status

1. Stage the index.html file.

git add Index.html

1. Commit index.html

git commit -m "Committing a Project File"

1. Make a few changes in index.html & create a new file info.txt file.

vi index.html

touch info.txt

1. Check git status. You should find index.html & info.txt as untracked files.

git status

1. Configure GIT to ignore all txt files.

Create a .gitignore file and then in that file add \*.txt and then save it. This will exclude all the files with the extension .txt

1. Again check the git status. You should find only index.html as an untracked file.

git status. Yes, as all the txt files are being ignored by .gitignore

1. Stage & commit index.html

git add . and git commit -m “Changes made in Index and Ignored all the txt files”

1. Log all your comments so far.

git log

1. Make some changes in index.html.

Added few lines of text in it using

vi index.html

1. Revert the change made in the previous step using git command.

git checkout -- .

1. Again change index.html.

Again added few lines of text in it using

vi index.html

1. Stage index.html

git add . to add the file in staging area and then git commit -m”Added few lines of text in it”

1. Revert back the last stage.

git rm –cached index.html

1. Rename ‘add’ command to ‘my-add’.

alias git my-add=’git add .’ to edit the name of the command using alias

Then source ~/project\_dir to save the changes in this particular directory

1. Using my\_add command Stage index.html again & commit the changes.

git my-add to stage the index.html file

1. Revert the last commit.

git reset –soft HEAD~1

*GIT Branching*

Objective: Commit HTML, CSS & JavaScript assignments into GIT.

SECTION-1 (HTML assignments) - Steps to follow:

1. First take a backup of your assignments & projects. This is required because due to incorrect GIT operation you may lose your files.

cp -r myrepo backup\_copy is the command used for then copy the folder to any other folder

1. Create an empty directory ‘Assignments’ & cd to ‘Assignments’.

mkdir Assignments

cd Assignments

1. Create a file README.txt inside ‘Assignments’ & write few lines about the contents of the ‘Assignments’ folder.

Create the file using this command ‘touch README.txt’ and the use command ‘vi README.txt’ which will open editor and then we can edit the file.

1. Commit README.txt file.

git commit -m “Adding new file README.txt”

1. Now create a new branch ‘html-assignments’.

git branch html-assignments

1. Switch to ‘html-assignments’ branch.

git checkout html-assignments

1. Copy all HTML assignments inside ‘Assignments’ folder.

Created two new files instead. assign1.html and assign2.html in the master branch

1. Commit HTML assignments into ‘html-assignments’ branch.

git add assign1.html

git add assign2.html

git commit -m “Committing assign 1 and 2 in branch”

1. Make minor changes into few files belonging to ‘html-assignments’ branch.

vi assign1.html With this command I made changes in the assign1.html file

1. Commit those changed files.

git add assign1.html

git commit -m “Modified assign1”

1. Switch to master branch.

git checkout master

1. Make minor changes into README.txt file & commit those changes into master.

By vi README.txt I edited the documents and then committed the changes by giving the command git add README.txt and git commit -m ”Changes made in README.txt”

1. Again switch to ‘html-assignments’ branch.

git checkout html-assignments

1. Make minor changes into few files belonging to ‘html-assignments’ branch.

By vi assign2.html I edited the document and then committed the changes by giving the command git add assign2.html

1. Commit those changes.

git commit -m ”Changes made in assign2.html”

1. Switch to master.

git checkout master

1. Merge ‘html-assignments’ branch into master. Confirm all html assignments are shown in master.

git merge html-assignments to merge the branches and git branch –merge to check the merged branches. Then opened the local repository to check the changes done.

Or we can check by running the command cat assign1.html ; cat assign2.html ; cat README.txt

Another way is to run the command git diff master..html-assignments

1. Finally delete the ‘html-assignments’ branch.

git branch -d html-assignments

SECTION-2 - (CSS assignments) Steps to follow:

1. Create a new branch ‘css-assignments’.

git branch css-assignments

1. Switch to the ‘css-assignments’ branch.

git checkout css-assignments

1. Copy all CSS assignments inside the ‘Assignments’ folder.

Created two new files instead. assign1.css and assign2.css in the master branch

1. Commit CSS assignments into the ‘css-assignments’ branch.

git add assign1.css

git add assign2.css

git commit -m “Committing assign 1 and 2 in branch”

1. Make minor changes to the README.txt file on line 1 belonging to the ‘css-assignments’ branch.

By vi README.txt I edited the documents and then committed the changes by giving the command git add README.txt

1. Commit those changed files.

git commit -m ”Changes made in README.txt”

1. Switch to master branch.

git checkout master

1. Make minor changes into the README.txt file on line 3 & commit those changes into master.

vi README.txt

git add README.txt

git commit -m “Committing README.txt”

1. Again switch to the ‘css-assignments’ branch.

git checkout css-assignments

1. Make minor changes to a few files belonging to the ‘css-assignments’ branch.

vi assign2.css

1. Commit those changes.

git add assign2.css

git commit -m “Edited assign2.css”

1. Switch to master.

git checkout master

1. Merge ‘css-assignments’ branch into master. Confirm all css assignments are shown in master.

git merge css-assignments

git diff master..css-assignments

1. Finally delete the ‘css-assignments’ branch.

git branch -d css-assignments

SECTION-3 - (JavaScript assignments) Steps to follow:

1. Create a new branch ‘js-assignment’.

git branch js-assignment

1. Switch to the ‘js-assignments’ branch.

git checkout js-assignments

1. Copy all JavaScript assignments inside the ‘Assignments’ folder

Created two new files instead. assign1.js and assign2.js in the master branch

1. Commit JavaScript assignments into the ‘js-assignments’ branch.

git add assign1.js

git add assign2.js

git commit -m “Committing assign 1 and 2 in branch”

1. Make minor changes into README.txt file on line 1 belonging to the ‘js-assignments’ branch.

vi README.txt then edited the 1st line

1. Commit those changed files.

git add README.txt

git commit -m “edited README.txt”

1. Switch to master branch.

git checkout master

1. Make minor changes into the README.txt file on line 1 & commit those changes into master.

vi README.txt then edited the 1st line

git add README.txt

git commit -m “edited README.txt”

1. Again switch to the ‘js-assignments’ branch.

git checkout js-assignments

1. Make minor changes to a few files belonging to the ‘js-assignments’ branch.

vi assign2.js

1. Commit those changes.

git add assign2.js

git commit -m “Edited assign2.js”

1. Switch to master.

git checkout master

1. Merge ‘js-assignments’ branch into master. Confirm all JavaScript assignments are shown in master.

To resolve the conflict, edit the README.txt file where the conflict is occurring then add it to the stage and commit it.

After this merge the branch using the command:

git merge js-assignments

git diff master..js-assignments

1. Finally delete the ‘js-assignments’ branch.

git branch -d js-assignments

*GIT Remoting*

Objective: Pushing source code into GITHUB & cotch\_llaborate with team members.

SECTION-3 (Pushing assignments to the remote repository) - Steps to follow:

1. Create a GitHub account if you do not have one already.

priyanshi-github

1. Login on into GitHub account.

Done

1. Create a new public repository ‘freshersbatch-oct16’.

Done. Freshersbatch\_Dec2021

1. Commit & push any sample file to this repository under the ‘Assignments’ directory.

git remote set-url origin <https://github.com/priyanshi-github/Freshersbatch_Dec2021.git>

git push -u origin

SECTION-4 (Pushing source code to a remote repository using Eclipse GIT plugin) - Steps to follow:

1. One developer from the project team will create eclipse projects ‘SampleProj’ & add sample source code files. Then commit all files through the eclipse GIT plugin.

One of our team member downloaded eclipse and created a project ‘SampleProj’ and added source code files named as sp1.html, sp2.css and sp3.js. She committed the code in the eclipse and pushed it in her GitHub account. **https://github.com/sayuyagami/SampleProj/tree/master/SampleProj**

1. Collaborate with other team members with your GitHub account so that they can also modify the committed files.

She collaborated with us by adding our GitHub accounts in the project. I was able to make changes in the files >> Adding and Deleting lines >> Committing them.

1. Other developers from same team will checkout all files from remote repository. This might get conflicts since certain files fail to merge. In such case, merge it manually.
2. Commit & push the ‘SampleProj’ project.

After committing I was able to push the changes that I made by clicking on the “Commit” button. Other collaborators were able to see the changes that I made and also I was able to see the changes that they made after committing and pushing the files.