GIT and GITHUB

→ maintaining
Saving the history of your project

→ At what particular time, which person made which change, where in the project. GIT helps in doing that

→ GITHUB is a platform, an online website that allows us to host our GIT repositories.

what is a repository?
- basically a folder where all the changes are saved.

cmd → (win) dir
git, LS (mac)

→ Terminal allows us to manipulate the file structure using commands.

1/11/23 DAY-1

GIT: is the version control system like tech & there are so many other online platforms that allow us to host these folders/ repositories or our project online so other ppl around the world can share look contribute & these are known as like Github, bitbucket, Gitlab.

cls → to clear screen in cmd

Terminal allows us to manipulate the file structure using commands.

commands: ls or dir (lists all the things in that folder)
mkdir (make directory / new folder)
cd (change directory)

→ where is this entire history being stored?
all of these ~~his~~ histories are stored in another folder that GIT provides us, this is known as a GIT repository & its named .git {name of the folder}
• dot in linux & mac OS systems, these are files that are hidden
now, how do we get this folder?
git init {initialise an empty git repository
how do I see that hidden file?
ls -a or dir /a
we can even see what's inside .git
command → dir .git

#commands: touch filename.txt {another way to mkdir}
if touch is not recognized use npm install touch-cli -g
rmdir filename {to delete ~~file~~ directory} use del {to delete file}
now git status tells the history but it is untracked & not saved
→ how do we maintain these changes?
git add . {all files / without '.' just specify file name puts everything on staye /staging area}
now they are ready to get pictures clicked
git commit {save updated file with changes or just save history}
notepad filename {to edit the file in notepad}

cat or type filename {to display the content of the file}

→ Suppose we don't want to take the picture of that person because it has already been taken or whatever like you don't want to commit ⇒ git restore --staged filename

→ now check if it is untracked/ removed from stage or not git status

git log {to see the history}

→ if you want to delete the commit just copy the hashID of the commit just below one like [commit 2, commit 1, commit 0] if you want to delete 1, 2 copy commit 0's hashID

git reset hashID

Now, it will be removed from git log

→ git stash {suppose we sent some ppl at backstage bcoz don't want to commit right now}

→ git stash pop {bring them back on stage}

→ git stash clear {removed from backstage also}

Now, I want my repository to be linked with my project —

git remote add origin url of repo

{means we're working with urls} → adding here, url → what is the name of url going to be that you're going to add (can be anything) origin is not keyword

git remote -v {all the urls that are linked to this project}

~~git~~ fork {to make changes in someone else's repo, makes a another version which is not going to ~~change~~ save changes in original repo}

git push {updates history on GITHUB}

git clone {downloads the project/repo}

Fork is the copy of that repo, it will not reflect changes in the main project until the project maintainer approves your changes and merge your code via a pull request.

git remote add upstream url {from where you've forked the project}

git branch Khushi { make new branch instead of committing in the main branch?

git checkout Khushi { head will now be on the Khushi branch { all the commits made now on will go onto the Khushi branch

git push ~~origin~~ new origin Khushi -f { force push? ~~when we have~~ removing a commit from pull request by force pushing to it. { we reset it to del that commit from local machine }

→ How to sync fork from cmd?

Step 1.) git fetch --all --prune
↳ all the branches ↳ the ones that are deleted

Step 2.) git reset --hard upstream/main { if doesn't work:-
git fetch upstream
{ fetch latest changes so that ur local repo is up to date }
git branch -r
{ lists branches available in 'upstream' remote. } }
reset the main branch of my origin to the main branch of upstream

To exit the "git log" ~~type~~ press "Q"

→ Rather than above two steps there's another way ~~to~~ ~~fet~~ :-
git pull upstream main { this only does it for local folders }
git push origin main { to fetch it on origin URL }

multiple commands can be written separated by ; semi colon
{ didn't work but GPT says use && instead of ; } I did it separately though (one-by-one)
&& worked.

Q. How do I merge several commits in a single commit?
→ git rebase -i hashID of last commit of origin or main something
this will open code editor
i → interactive shell

Pick & Squash
pick → means you're taking this commit
s → above this s whichever pick you have, merge your commit in that

Ex- pick 1
s 2
s 3 } will be merged into 1
pick 4

save the file
press ctrl+w (win) or ESC :x (mac)

§ delete other messages that you don't want §

Merge conflicts →