***Report- Git and GitHub:***

What is Git:

Git is essentially to store the different versions of the code you’re writing which can be viewed and accessed. It is like how history works.

How to use Git:

To use git, you must first download gitbash. Git commands are written and from terminal. Here are a few git functions and what they do-

* ls – list all the files and folders in the current folder.
* cd – change directory, changes the directory based on the address enter following the command. Syntax – cd ~” file location”
* git init – used to create git file which tracks all changes in the current folder.
* Git status – shows all changes since previous commit.
* Git add <file/folder> - preps the files or folder for committing. By using “.” After “add” you can add all files and folders in the current folder at once
* Git commit -m “msg” – a commit is essentially the version of the prepped files. This function pushes the files to the git file. The msg displays so that we can understand which version it is. We can also modify any errors in our previous commit instead of creating a new one if we add “--amend” after msg.
* Git log – this shows us the commit history in the branch that you are currently in. Starting from the latest commit in reverse chronological order. If we add the “--all” after log it will show us the commits from all the branches and we can further and “--graph" to better improve our readability.
* Git reset <file/folder> – this function essentially performs the opposite task of git add. It removes the files that are prepped for a commit.
* Git checkout – this command is used in case you want to view one of the previous versions of your code. To do this we must all and the branch\_name or commit\_hash. You can also restore the contents of a previous version using this command by simply adding the file/folder you want to restore after the commit\_hash.
* Git config --global alias.shortcut <command>
* .gitignore – this creates a file whose contents git does not track.

What is GitHub:

Github is an online site similar to that of google drive or iCloud. However, unlike these github is specifically designed for git. It is used to create a remote backup of all your files tracked by git.

Git is a local repository while Github is a remote repository.

How to use Github:

* First create a remote repository.
* Now to link the remote repository to our local repository.
* Git remote add <remote\_repository\_name> <url> - this command allows us to link our local repository to our remote repository. The remote\_repository\_name is usually kept as origin.
* Git remote – this command gives us a list of all the remote repositories linked to our local repository. We can also get more details by adding -v to the existing command.
* Git remote remove – removes the link between the local repository and the remote repository.
* Git push < remote\_repository\_name> <branch> - uploads the version history of the branch git was tracking to remote repository. If there is already something saved on the remote repository and you wish to overwrite it then you can add -f to the command. This causes the git to force push the branch.

To download code from github:

* git clone <url> - downloads the remote repository in url to your computer.
* Git fetch – this command updates the remote tracking branches and is also used to view what are all the changes in the remote repository. This does not change your current working directory like git pull, it is used only to view.
* Git pull <remote\_repository\_name> <branch> - updates the local branch with the changes made in the remote repository.
* There is also a shortcut than can be used with both push and pull - Git push/pull <remote\_repository\_name> <branch> --set-upstream. This function makes it so that using git push/pull automatically performs the task to <remote\_repository\_name> <branch>.
* Git branch <branch\_name> - creates a copy of the existing branch which can be worked with separately. Used to design new feature to existing software.
* Git checkout <branch\_name> - used to switch to the <branch\_name> and start working on that branch.
* Git branch -D <branch\_name> - deletes the branch.
* Git merge <branch\_name> -m “msg” – used to merge <branch\_name> with the current branch (the one HEAD is pointing to). the result of a merge will save the changes from both branches. The result of the merge gets saved to the current branch as a commit.
* Merge conflicts – arises when you try to merge two branches where the code in the same lines is changed. Git does not know which change to keep and will therefore give us prompt asking us which of the changes we want to keep. Remove all the lines of code baring the line we wish to keep and run “git add” and “git commit”.

Feature branch workflow:

A popular method companies use to add new features.

1. Create branch for new features (use previously mentioned commands)
2. Upload the feature branch to github.
3. Create a pull request on github (this allows your teammates to do code reviews and add comments
4. Merge the feature branch to main on github by clicking on merge pull request.
5. After merging update local repository to make sure everything is upto date.
6. Merge conflicts in feature branch workflow can be solved either on your computer and then uploaded to github or you can directly resolve the issue in github. The method, however, remains the same as before that is to keep only the lines of code we wish to have and delete the rest.

References:

<https://youtu.be/hrTQipWp6co?si=fjy6mAD7L50hvl-H>

<https://youtu.be/1ibmWyt8hfw?si=gljNBb4HbjRKuWhu>

<https://youtu.be/Q1kHG842HoI?si=9Vja9qgxeIVVWDov>