**Lab No: 02 GIT & GitHub**

**PART 1: GIT**

Installing GIT and checking version:



Setting username, e-mail, and editor and checking the list of details:

A screen shot of a computer screen

Description automatically generated

Created a directory and navigated into it:



Making my directory a git directory:

A screenshot of a computer program

Description automatically generated

**Q) Which directory git uses to track all changes? Run the command for it.**

**Ans:** Git used a directory called **‘git’** within its project directory to track all changes and store repository information.

A black screen with white text

Description automatically generated



A screenshot of a computer

Description automatically generated

First\_file.txt created and written the contents in it:

A screen shot of a computer screen

Description automatically generated

Checked the status of git:

A screen shot of a computer

Description automatically generated

Two theory questions follow:

**Q) Explain the output?**

Ans: When I executed the command ‘git status’ it shows that I am on the main branch of the git and have not committed any files to the branch though I have created one (first\_file.txt). So, it is showing a message to add the file to git for tracking.

**Q) Explain the three-tree architecture in Git?**

Ans: In Git, there are three main layers:

**1. Working directory:** This layer is created when you initialize a Git project on your local machine. It allows you to edit and make changes to the source code.

**2. Staging area:** After making edits in the working directory, you can stage those changes using the "git add" command. This step acts as a preview for the next stage and allows you to choose which changes to include. If further modifications are made in the working directory, the snapshots for these two layers may differ, but they can be synchronized by using the same "git add" command.

**3. Local repository:** When you're satisfied with the changes and want to save them as a permanent snapshot, you can use the "git commit" command. This action replicates the latest snapshots from all three stages, ensuring they are in sync with each other and creating a new commit in the local repository.

Added the file in git and committed the changes:

A screen shot of a computer

Description automatically generated

Verifying the commit that has been made:

A screen shot of a computer

Description automatically generated

Two theory questions follow:

**Q) Explain in detail what is SHA value in Git and what is the SHA value in your previous output?**

Ans: In Git, a SHA value is a 40-character hexadecimal string employed to uniquely distinguish an item within a Git repository, whether it's a commit, tree, blob, or tag. This SHA value is generated by applying the SHA-1 hashing algorithm to the item's contents. The significance of the SHA value lies in its role within Git's content-addressable filesystem. This means that Git identifies items in a repository based on their content rather than their location. Consequently, it becomes straightforward to relocate or duplicate items between repositories without disrupting their links.

The SHA value in my previous output is **‘992c257a1a01a50a3669e8c1e72cb14490ad1113’**.

**Q) Explain what HEAD is and what is a master?**

Ans: In Git, the HEAD is a pointer that directs you to the presently active branch, typically the default "master" branch. When you create a new commit, Git simultaneously updates both the "master" branch and the HEAD to point to this latest commit. This action effectively advances them along the commit history. Consequently, the HEAD consistently reflects your current position in the branch you're working on, while the "master" branch records the most recent commit in that branch.

In Git, the "master" branch functions as the primary and default working line in a repository. It operates as a dynamic indicator, continually directed at the most recent commit in the project's timeline. Consequently, "master" symbolizes the most up-to-date and reliable rendition of the codebase, serving as a reference for ongoing development and as a basis for initiating new branches or implementing new features.

Checking where the head is pointing:

A screen shot of a computer

Description automatically generated

Creating a second file and committing the changes:

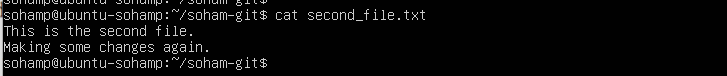
A black screen with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

Again, making some changes to the file:



Checking the changes that has been made:

A computer screen with white text

Description automatically generated

The theory question follows:

**Q) Explain the output seen in above SS? – anything written about GIT DIFF command.**

Ans: The "git diff" command when executed, it performs a comparison function on various Git data sources, including files, branches, commits, and more. Its primary purpose is to display the differences or changes between these data sources, such as changes between commits or the differences between a commit and the current working tree. Essentially, it helps users track and manage modifications in version-controlled files and resources.

In my screenshot, it is comparing a/second.file.txt which was the original state of the file with the modified version of the file b/second.file.txt. It shows that the line “Making some changes again” has been added. This is the reason why “++” symbol is before that line and the line is highlighted in green.

Made some changes to the first file and checking the differences in all the files:

A screenshot of a computer program

Description automatically generated

Checking the differences that has been made in the first file:

A screenshot of a computer screen

Description automatically generated

**PART 2: GITHUB**

Creating a GitHub account:

A screenshot of a computer program

Description automatically generated

Created a new repository of named smu-eets8357-SP

A screenshot of a computer

Description automatically generated

Cloning the git directory in local machine:

A screen shot of a computer

Description automatically generated

Creating an index.html file and ran to see the below output:

A blue background with black text

Description automatically generated

Added an image:

A red and blue flag with a horse

Description automatically generated

Reverted back to the original file without image:

A blue background with black text

Description automatically generated

HTML code:

A screenshot of a computer program

Description automatically generated

Creating Personal access token:

A screenshot of a computer

Description automatically generated

The theory question follows:

**Q) What is a branch, and which is the default branch?**

Ans: Branches allow us to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository.

When we set up a GitHub repository with content, it comes with a single branch, which GitHub designates as the default branch. This default branch is the one shown to visitors when they access your repository and is also the initial branch checked out by Git when someone clones the repository. It serves as the base branch for new pull requests and code commits unless another branch is specified. By default, GitHub names this primary branch "main" in new repositories.

Creating a new branch off master and named the first-branch, checking the current branch and seeing the location of first-branch:

A black screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

Checking the pointer for HEAD:

A screen shot of a computer

Description automatically generated

**Q) Is the head different or the same? Explain your output.**

Ans: When we create the first-branch the branch gets created and we do not shift to the first-branch. The HEAD remains the same as I have not shifted to the new branch. So, the head is pointing to the main branch which is the default branch in the repository. The ‘\*’ represents the current branch in which I am currently in.

The changes has been pushed to the remote directory:

A screenshot of a computer

Description automatically generated

Adding the url <https://www.smu.edu> in index.html:

A screenshot of a computer program

Description automatically generated

The changes has been reflected in the index.html file:

A blue background with black text

Description automatically generated

Created a new branch named the second\_branch and switched to that:

A screen shot of a computer

Description automatically generated

Created two HTML files in first-branch and pushed it to remote directory:



A computer screen shot of white text

Description automatically generated

A screenshot of a computer

Description automatically generated

Created one HTML file in second-branch and pushed it to remote directory:



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Merging second-branch with main branch:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

After merging, running the command to check where is head pointing and checking whether the branch has been merged:

A screenshot of a computer screen

Description automatically generated

Created a new branch called third-branch and switched to that:

A screenshot of a computer

Description automatically generated

Now editing any one of the files in the repository and commit it in current third\_branch.

**Switched to the master branch and check for the changes you made. Was it there? If not, explain.**

Ans: The changes will not be there, as I have not merged the third branch and the master branch. They are separate.

Go to the same file again and make different changes and commit it.

Initial file:

A screenshot of a computer

Description automatically generated

After edit:

A screenshot of a computer

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

In main branch the file did not get updated:

A screen shot of a computer

Description automatically generated

Making different changes to the file in third branch:

A screenshot of a computer screen

Description automatically generated

Now merging third-branch with main

A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

**Was it successful?**

Ans: Yes, because when I have merged the updated contents of the third-branch gets reflected in the main branch.

Now the updated html came in main branch:

A screenshot of a computer

Description automatically generated

Create a repository with three branches (1, 2, 3), and file-a exists in all three branches. File-b exists in branches 1 and 2, and file-c exists in branch-3. **Explain the behavior/outcome if you switch to any of these branches and then delete the local git repo.**

Ans: The system will not allow me to delete the repository from any of the branches because I am already inside it and from within a branch, I cannot delete the root. So, to delete the repository I will have to come outside of the directory and then delete it.

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A black screen with white text

Description automatically generated

\_\_\_\_\_END\_\_\_\_\_