**COURSE CODE:** DITL803 **DATE:** 21/1/2022 **COURSE NAME:** DevOps Laboratory **CLASS:** BEIT

**EXPERIMENT NO.01**

**LO:**

1. To understand the concept of DevOps with associated technologies and methodologies

2. To understand different Version Control tools like GIT, CVS or Mercurial

**AIM / OBJECTIVE:**

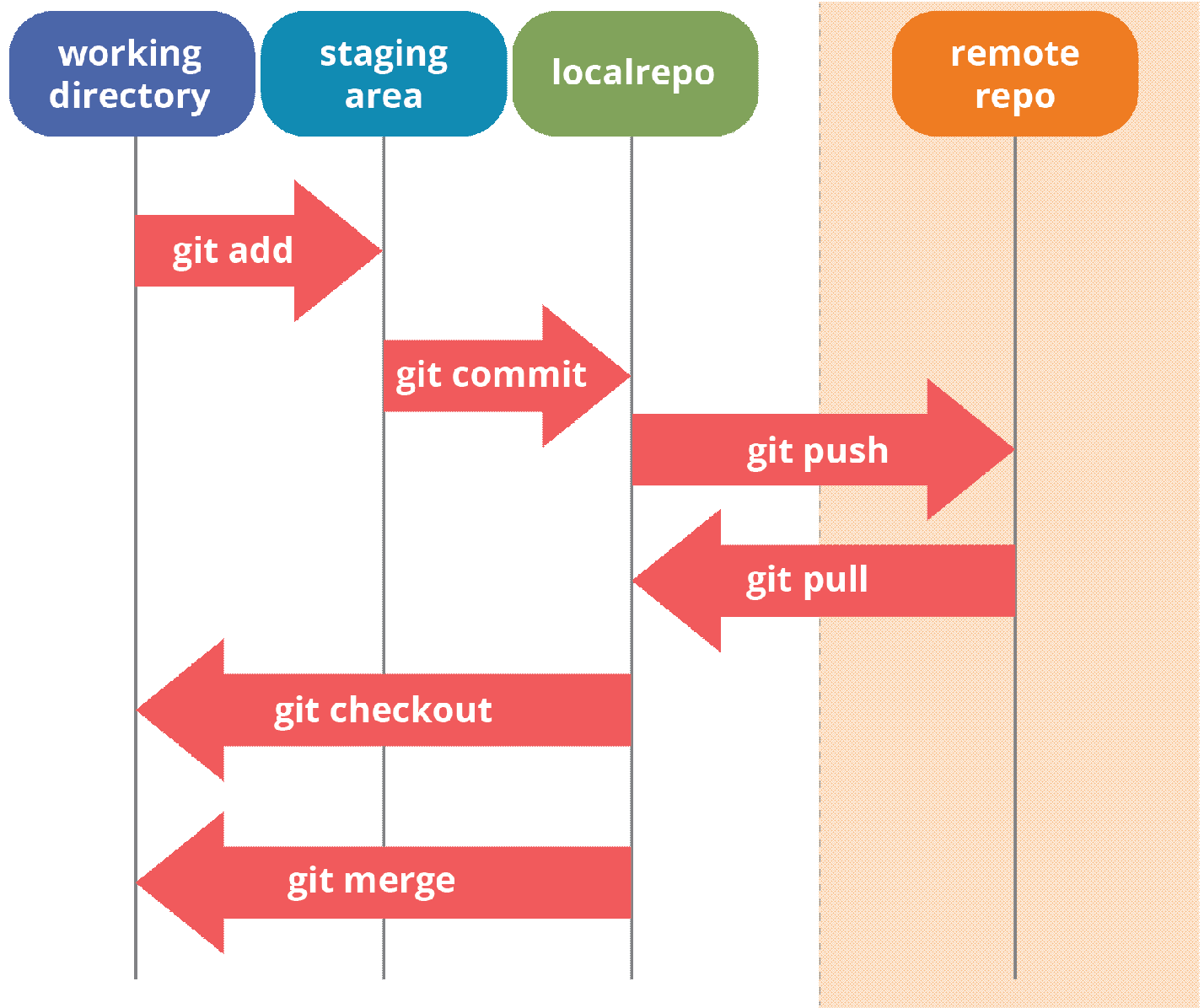
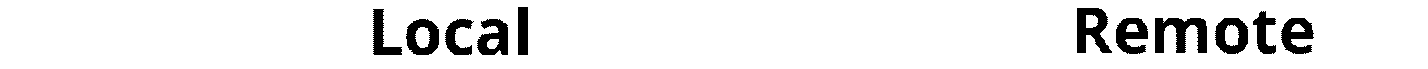
To Perform Version Control on any project using any Version control tool (GIT). To study various commands to install and organize you files in the repository and their implementation. Commands for understanding/ implementing Git version, set configuration variables, make new directory & access it, start a new project with Git, Git status and Git add, commit, log, clone, Pull and Push.

**THEORY:**

* Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
* Git is easy to learn and has a tiny footprint with lightning fast performance.
* Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.
* It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.
* Some of the basic operations in Git are:
  1. Initialize
  2. Add
  3. Commit
  4. Pull
  5. Push
* Some advanced Git operations are:

1. Branching
2. Merging

* The following diagram depict the all supported operations in GIT



**Task1 : Installation of GIT**

* Installation of GIT can be done in both Windows and Ubuntu Operating Systems.
* The installation process is as follows:

# Windows Operating System:

* Download GIT from <https://git-scm.com/> and perform straight forward installation.

# Ubuntu

* $sudo apt install git

# MacOS,

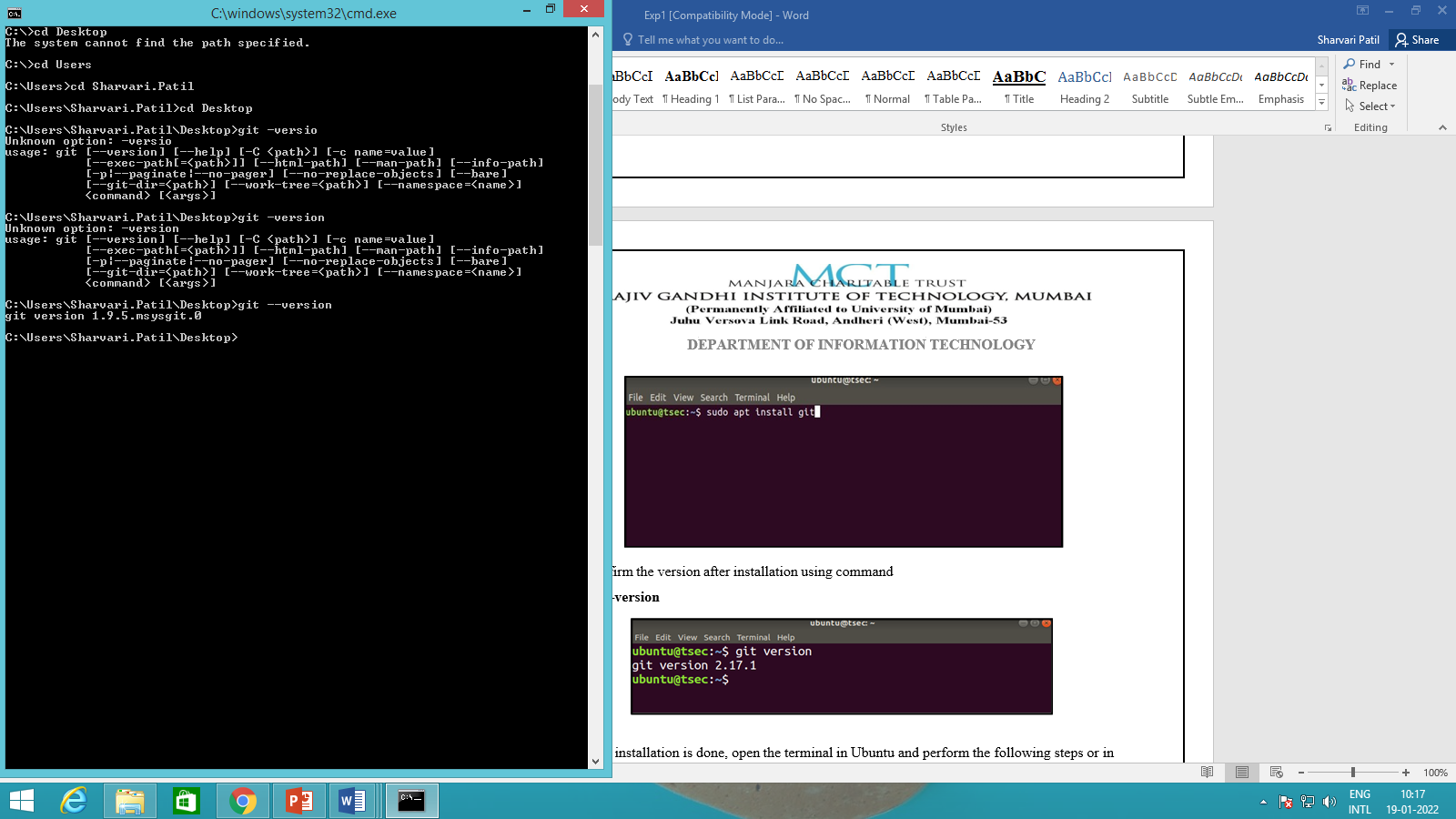
* download git from <https://git-scm.com/download/mac> or download Xcode command line tools.

**Task 2 Confirm the version after installation using command**

Command :

1. **$git --version**

O/P :

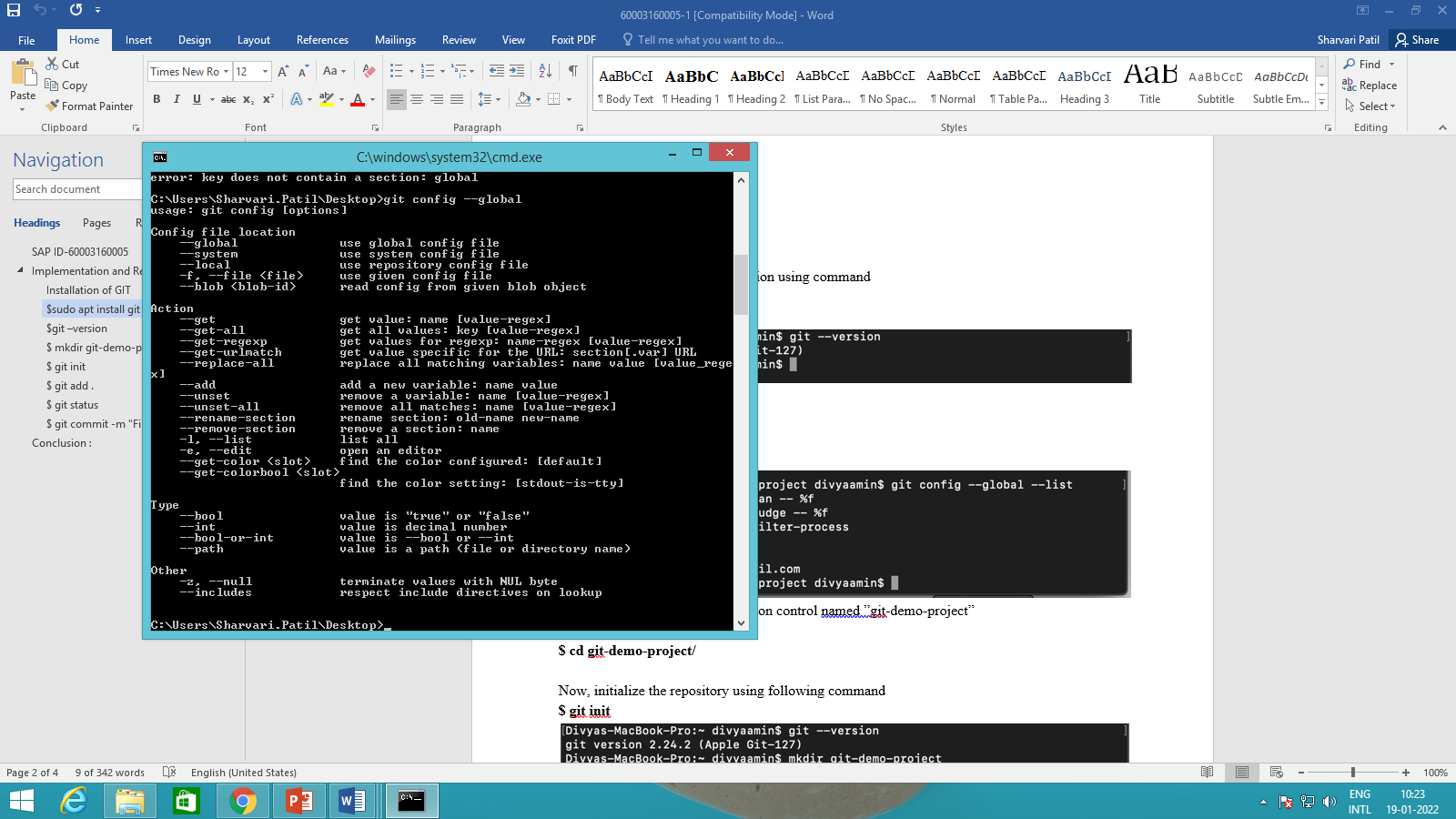


1. $git –config

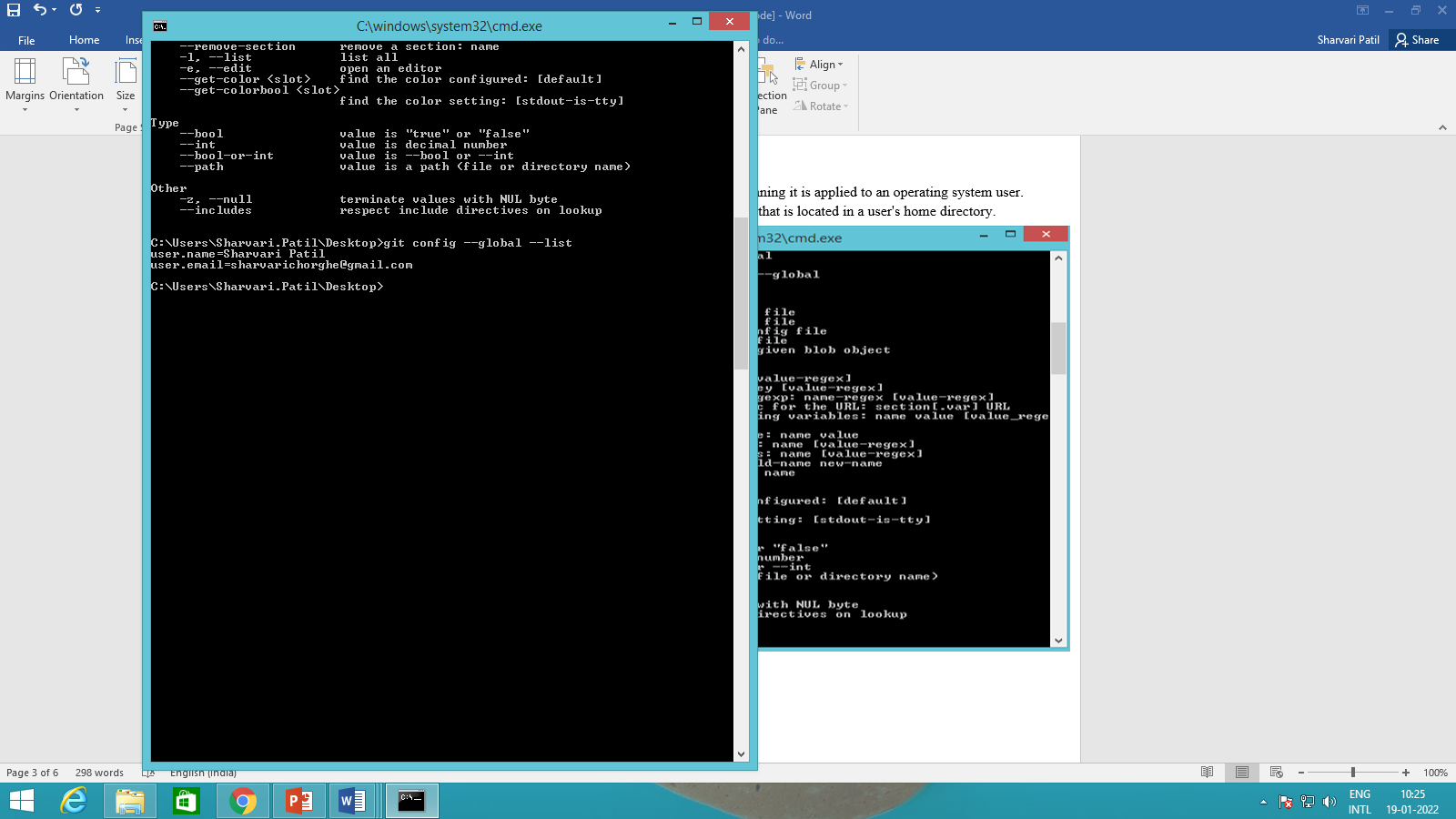
The git config command is a convenience function that is used to set Git configuration values on a global or local project level.

$git –config --global

Global level configuration is user-specific, meaning it is applied to an operating system user. Global configuration values are stored in a file that is located in a user's home directory.

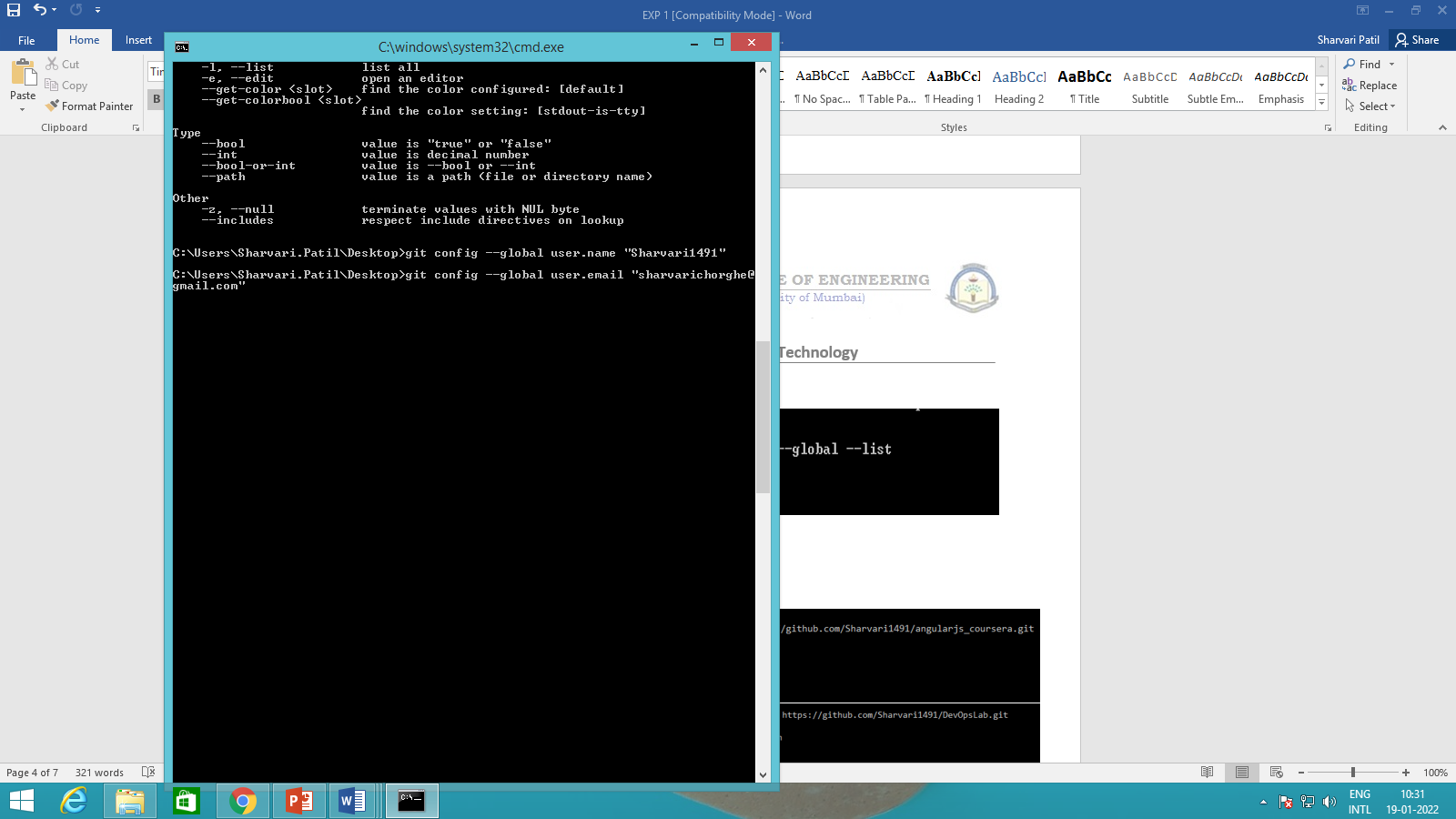


1. **$ git config --global --list**



# $ git config --global user.name "<user\_name>"

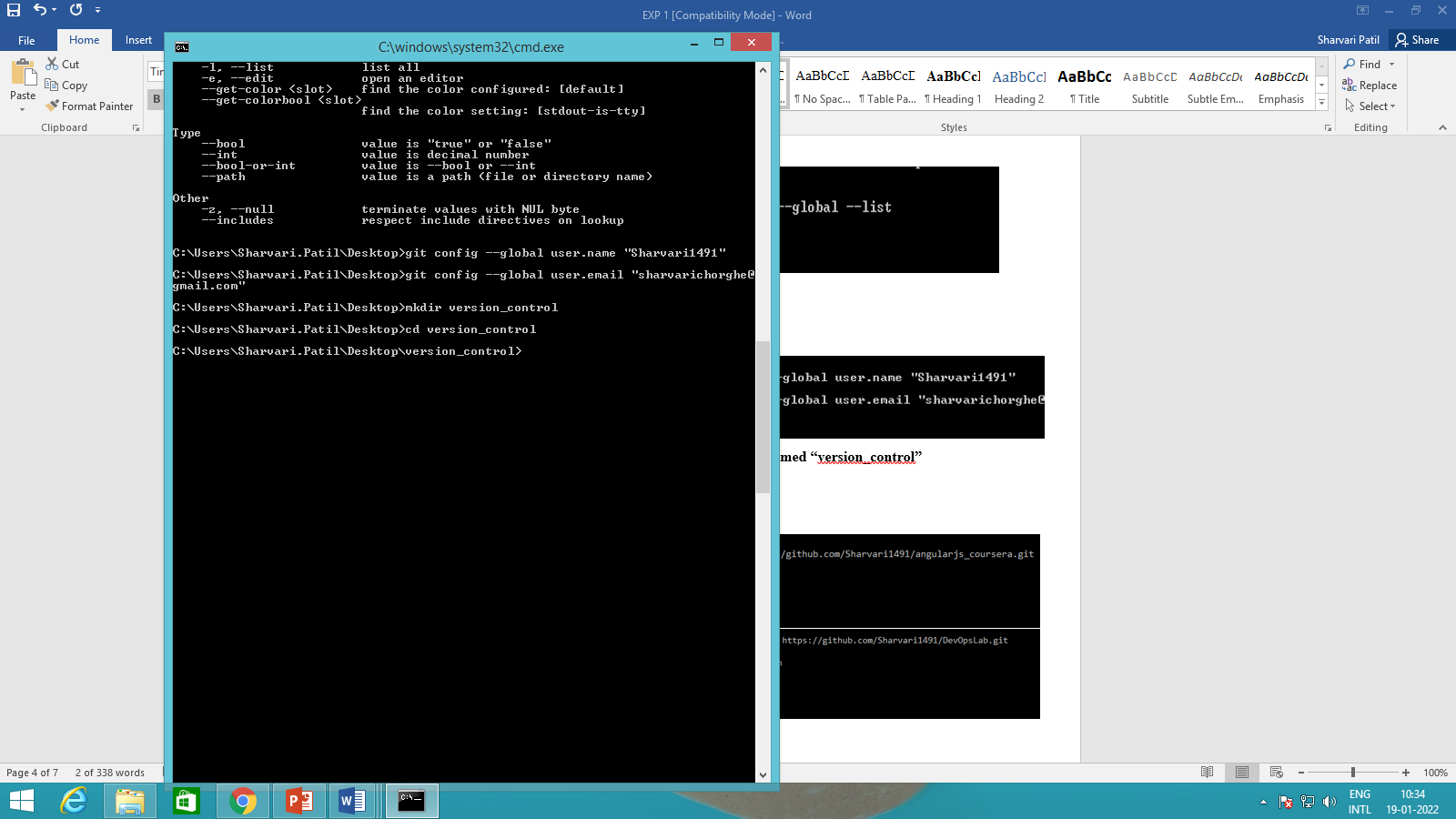
**$ git config --global user.email "<user\_email>”**



1. **Let us create a repository for version control named “version\_control”**

**$ mkdir version\_control**

**$ cd version\_control**



1. **initialize the repository using following command**

**$ git init**



1. **Now, let us add some files inside our repository “version\_control”**

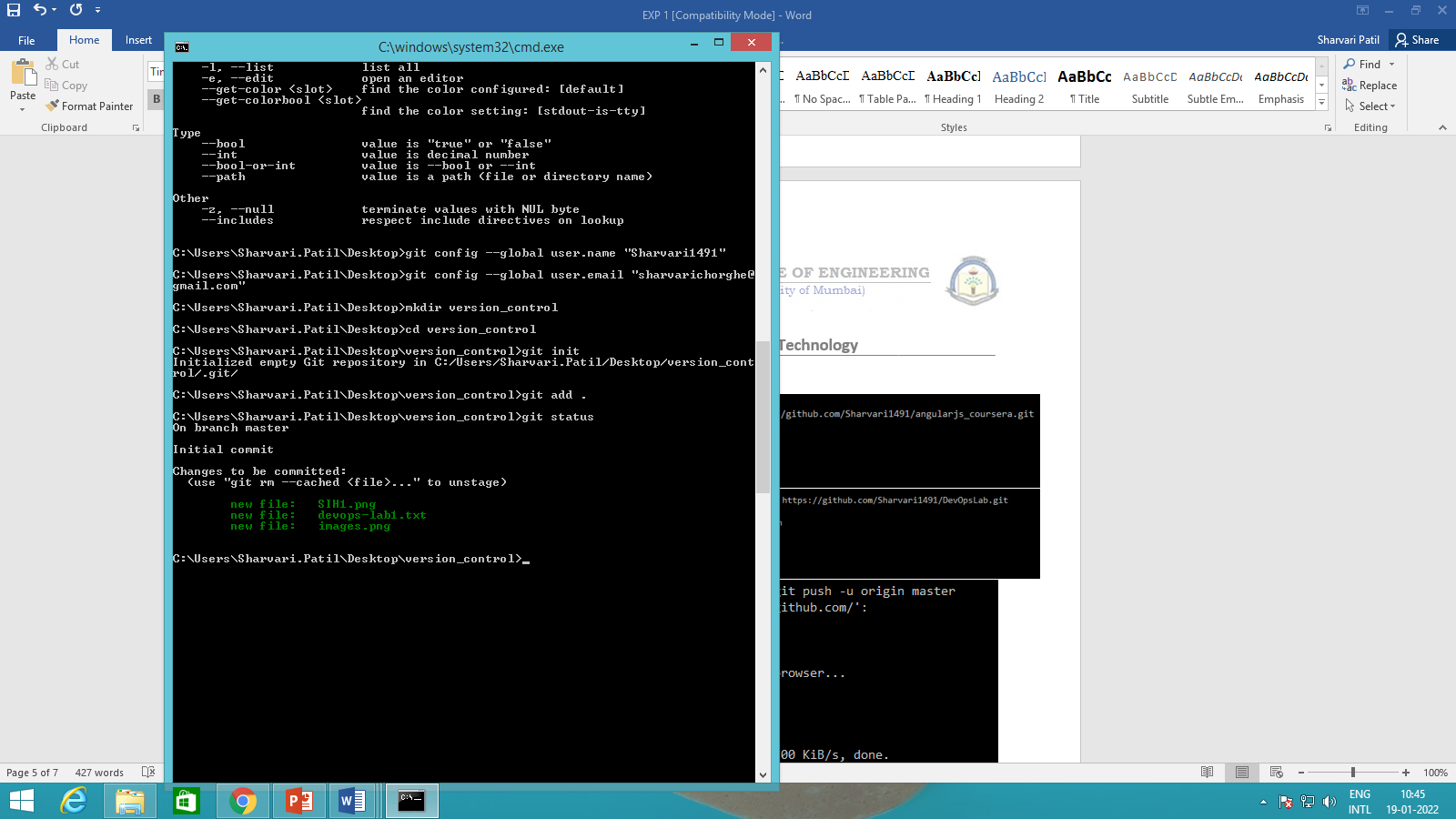
* To add files in the repository by create or copy some doc,html,image files inside current directory to see index and staging area.
* The add command is used along with dot (. Dot means current directory) for adding files in current repository i.e. making them in staging mode. They are untracked until we commit them.

**$ git add .**

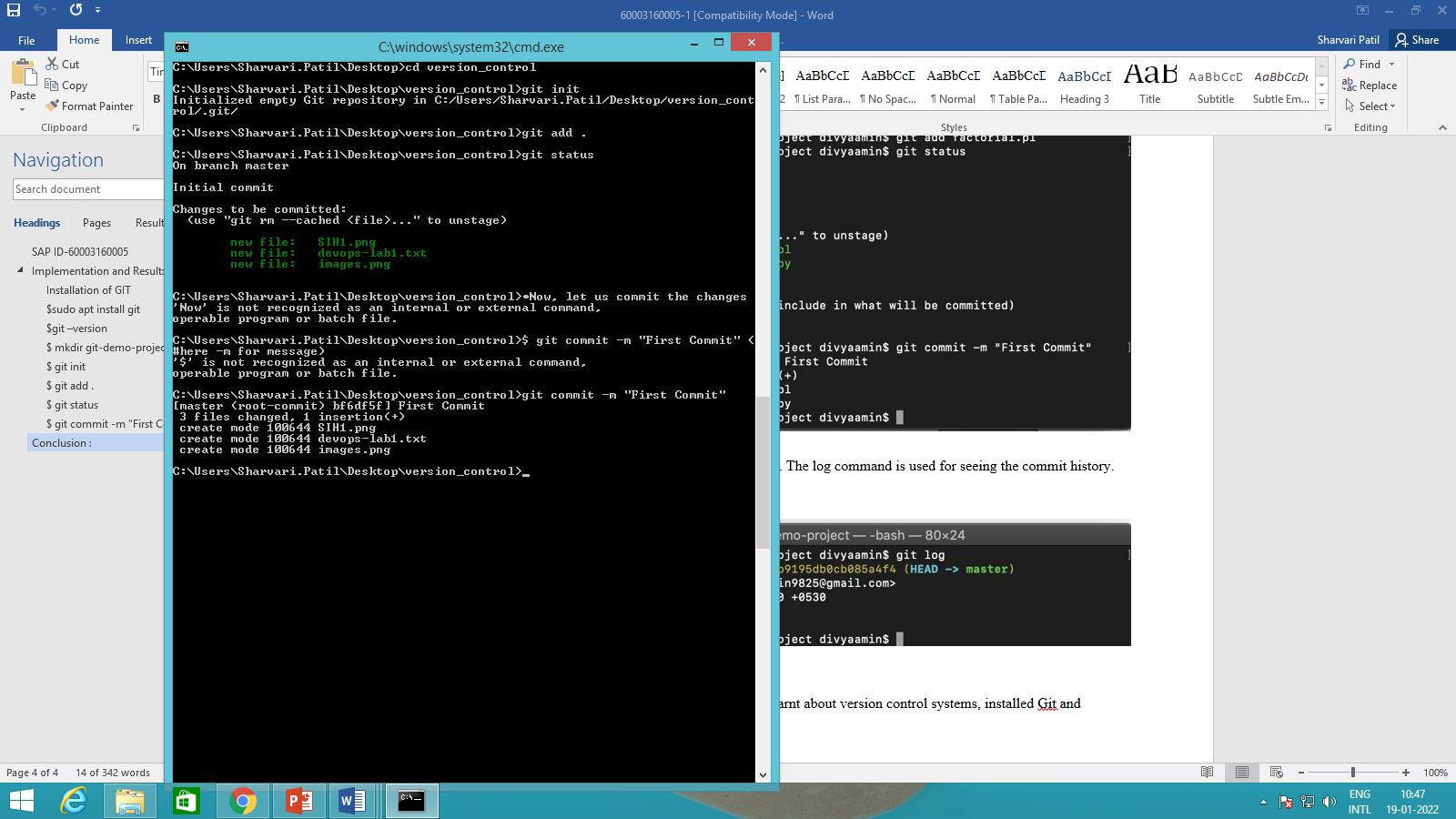
* To check the status of repository, use

**$ git status**

* Now, let us commit the changes

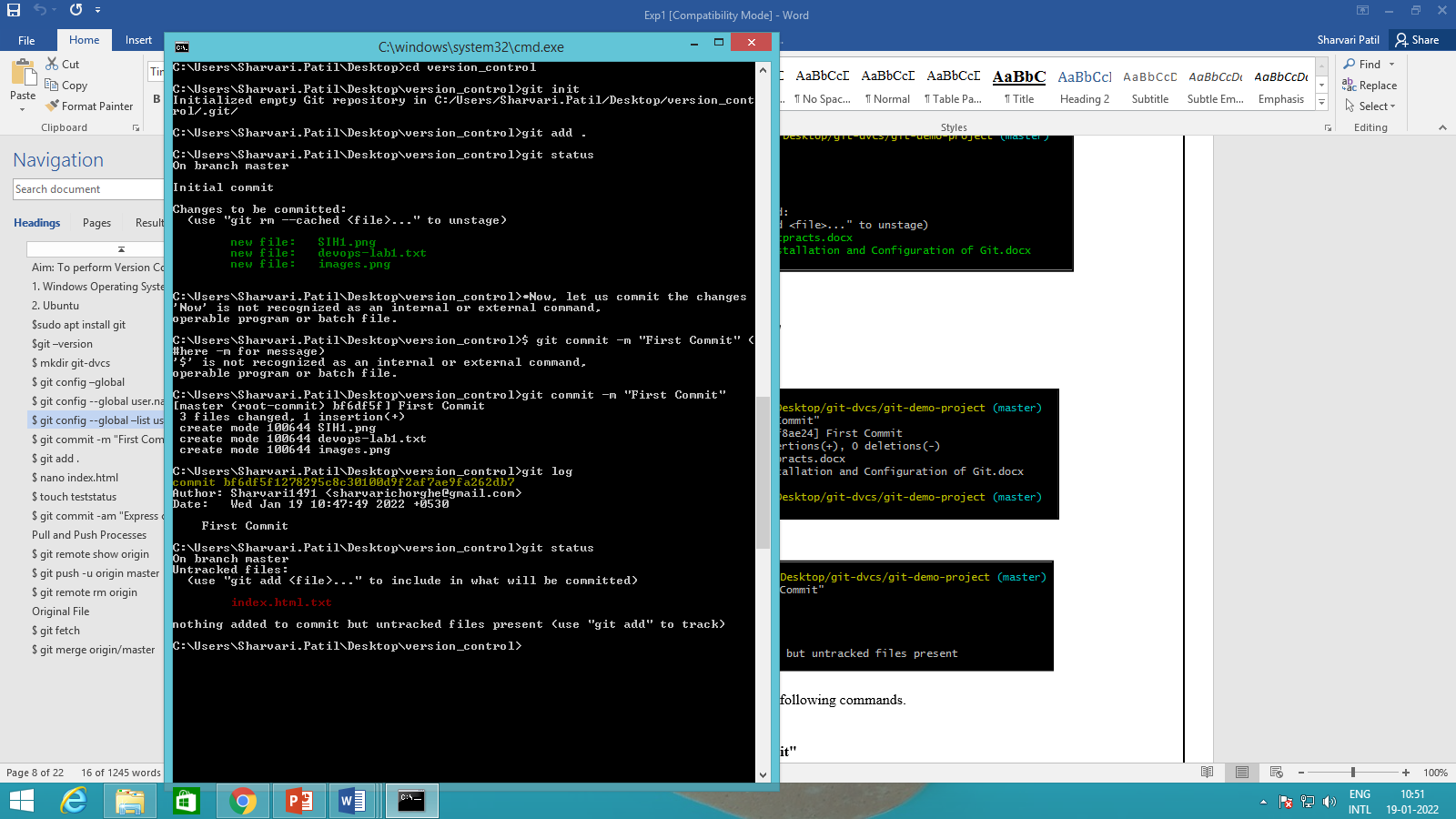


# $ git commit -m "First Commit" (#here -m for message)

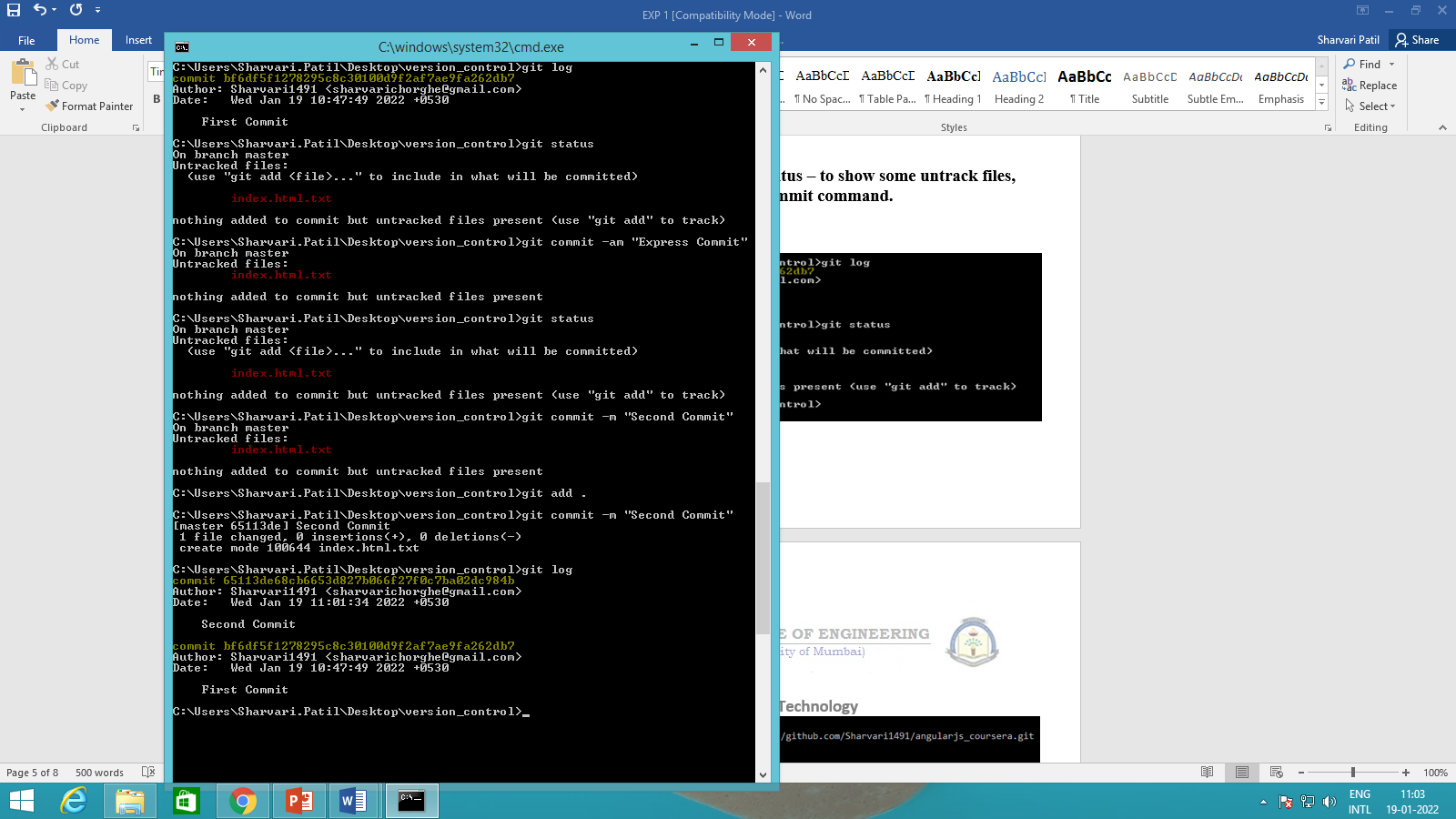


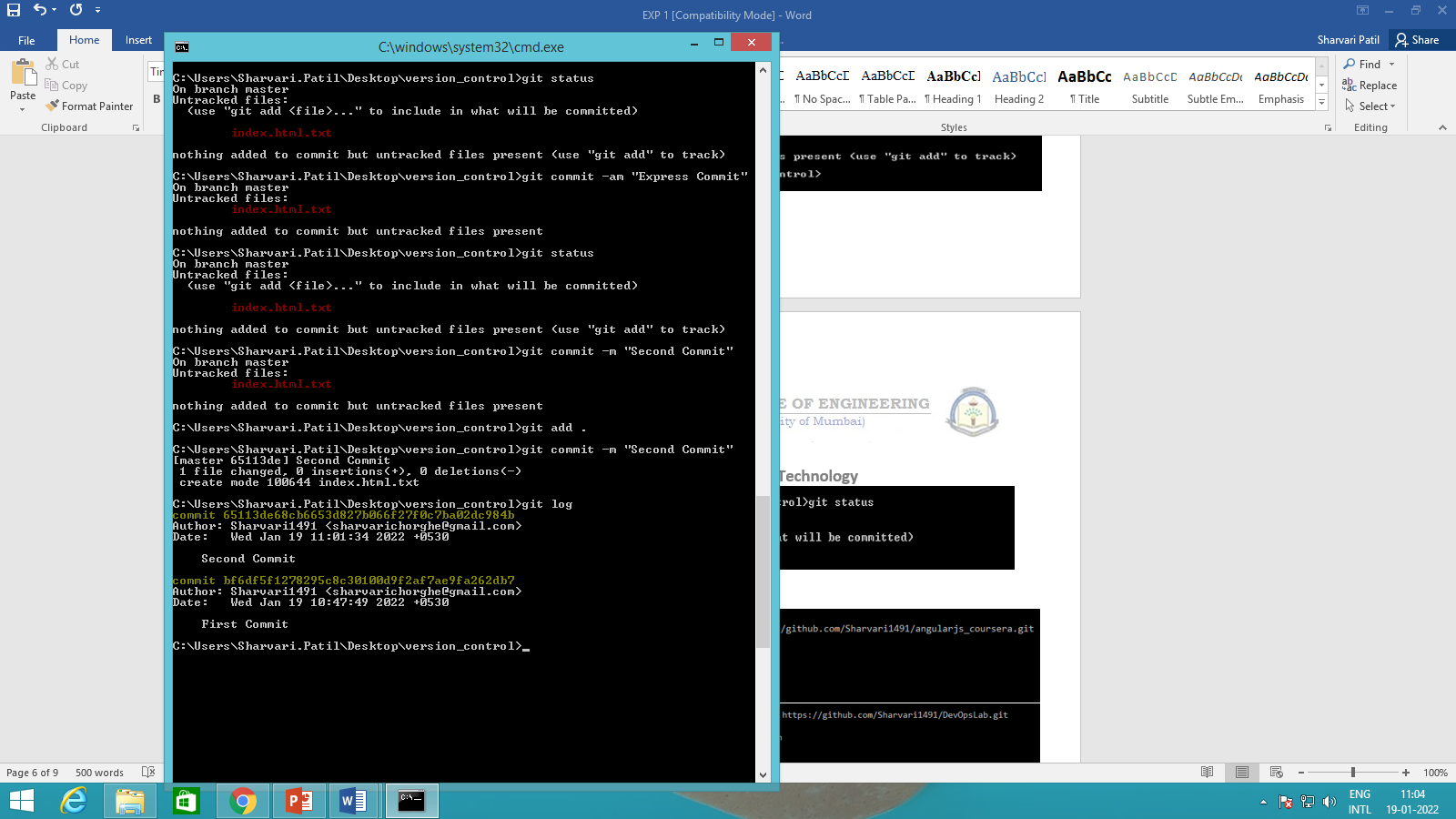
1. **$git log- to check the history and $ git status – to show some untrack files, so untracks files can be tracked using commit command.**

Add index.html in our directory



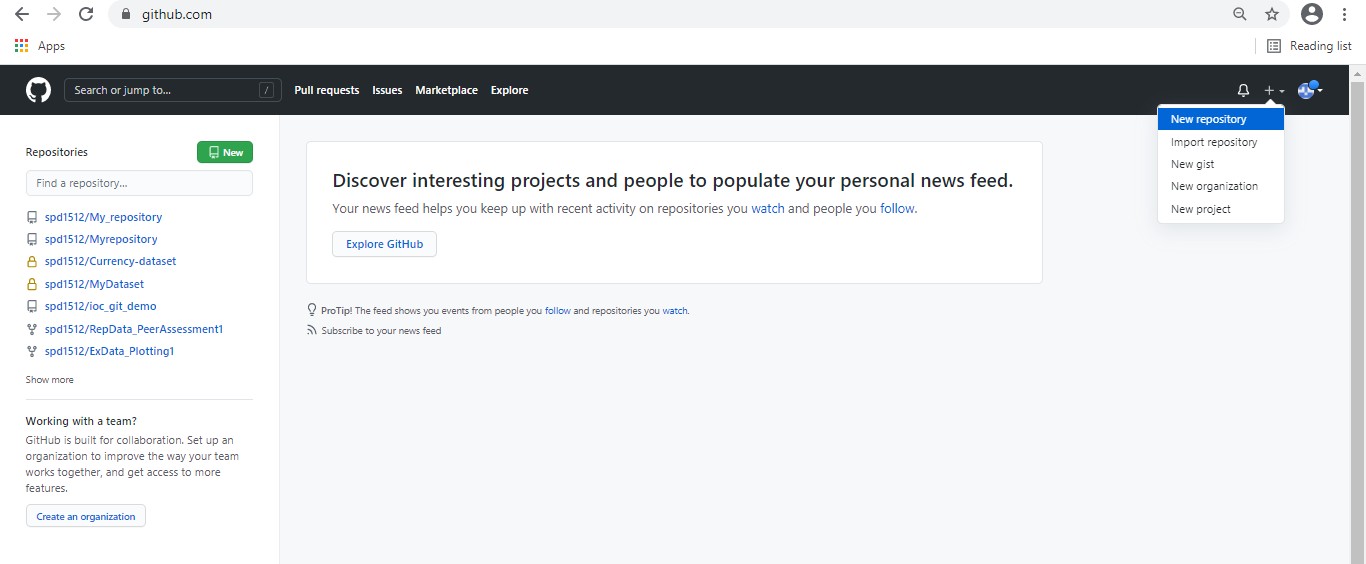
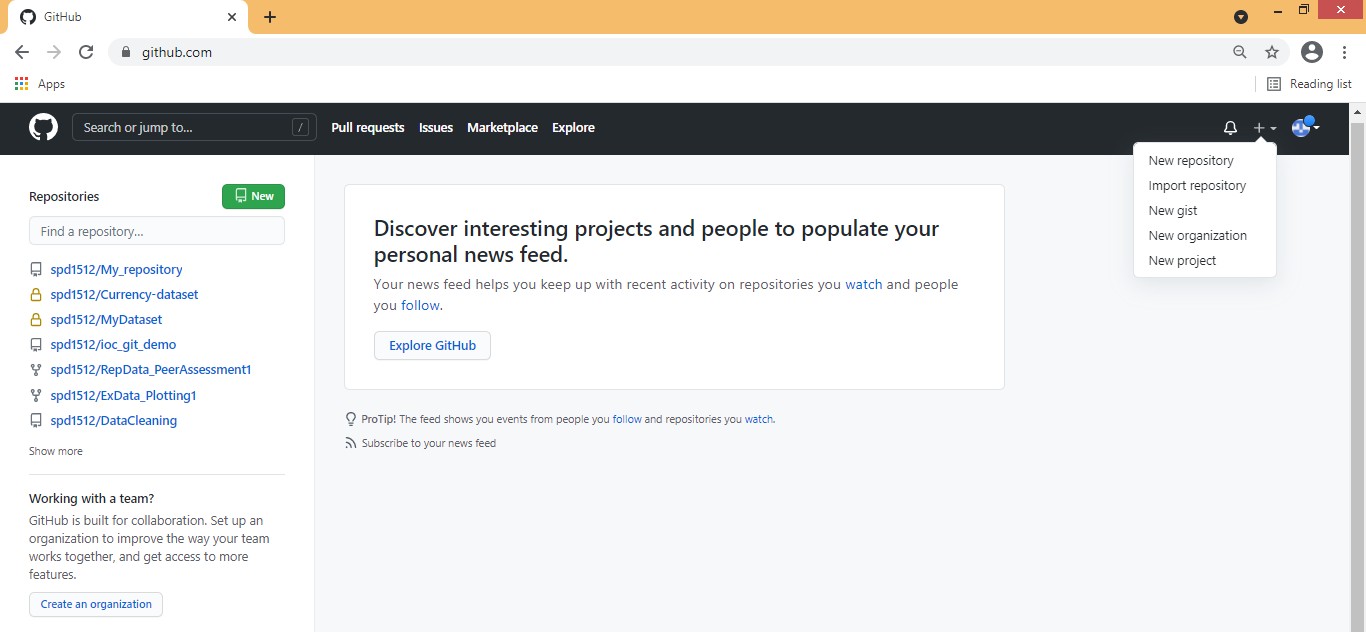
1. Commit the new file added.



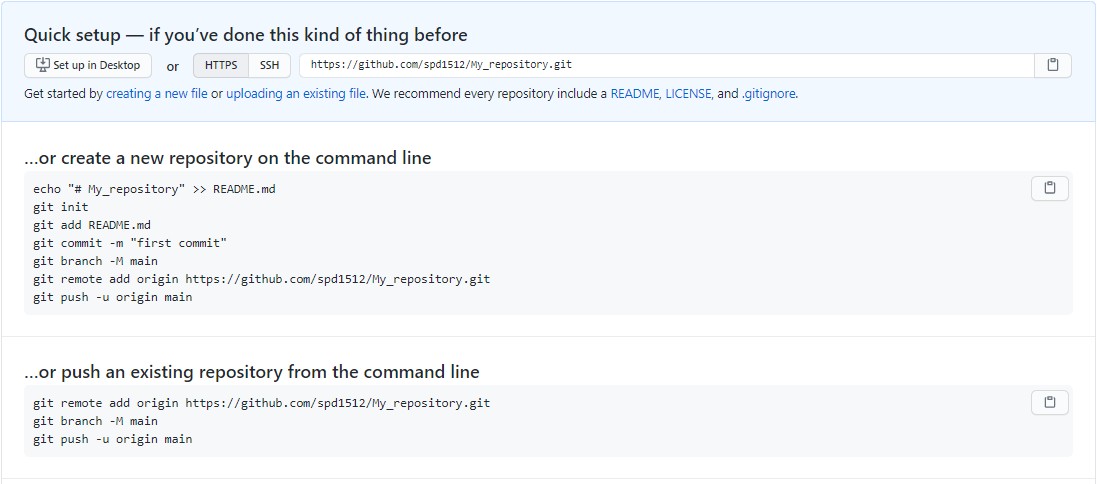
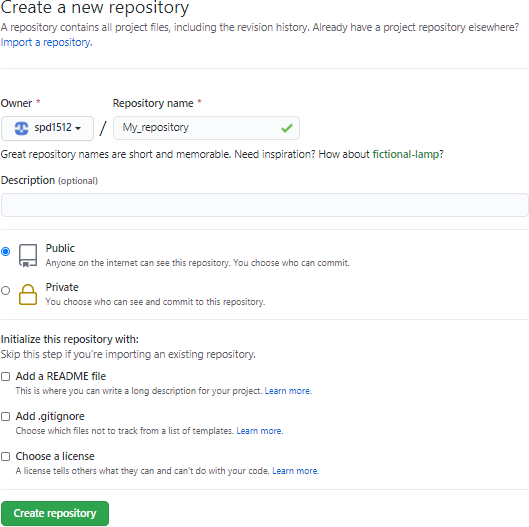


**GitHub Integration with pull push remote origin fetch and merge commands.**

* Open Github in desktop
* Log into your Github Account
* The following screen is displayed:
* Select the “ New Repository” option

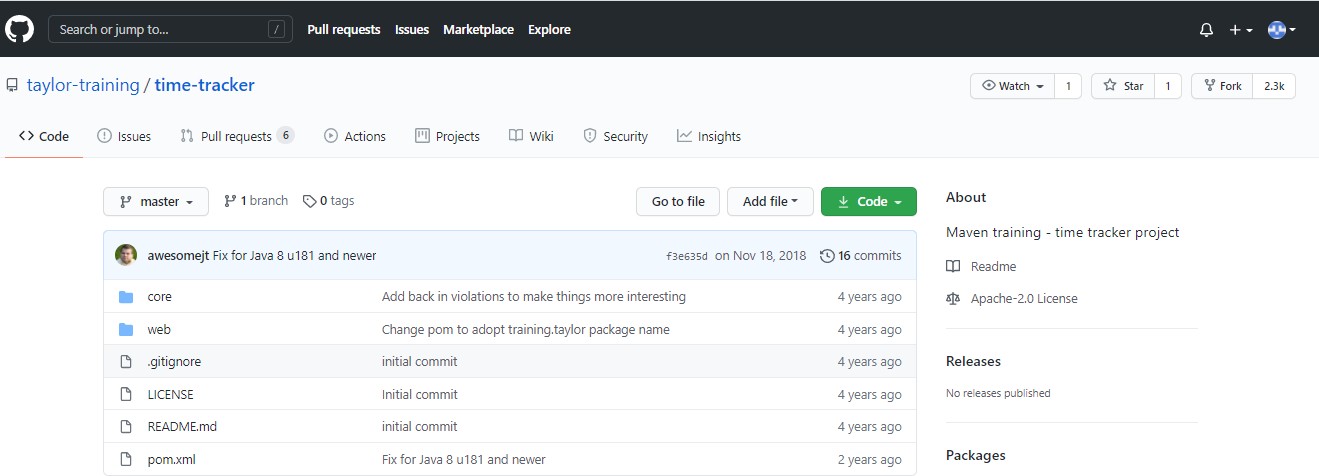


* Now Specify a Name to repository and select public option followed by create repository.

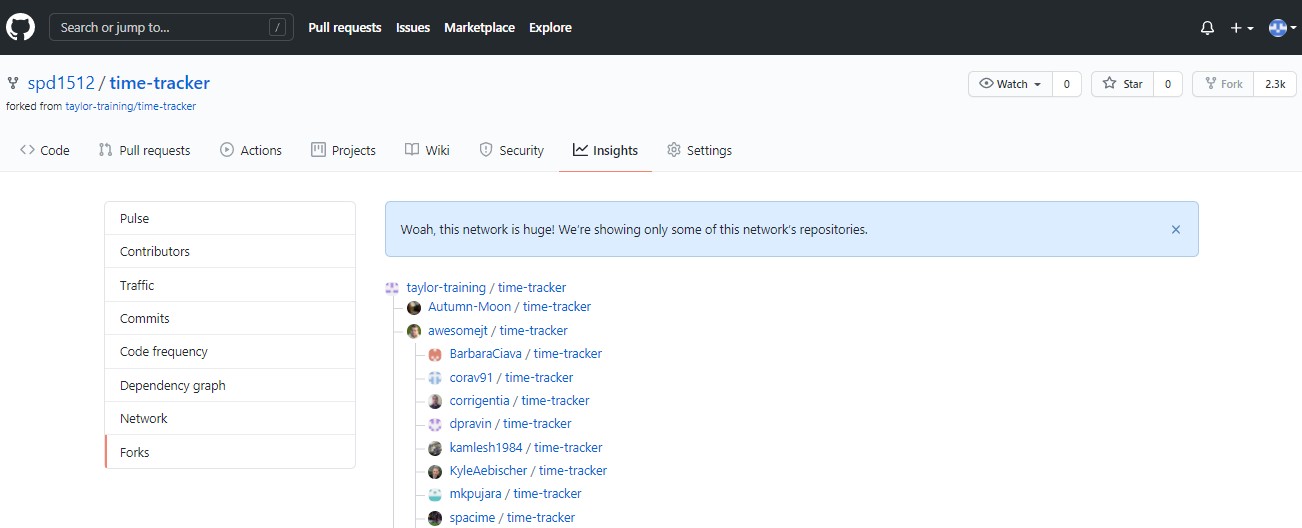


**Fork Operation and Delete a Repository**

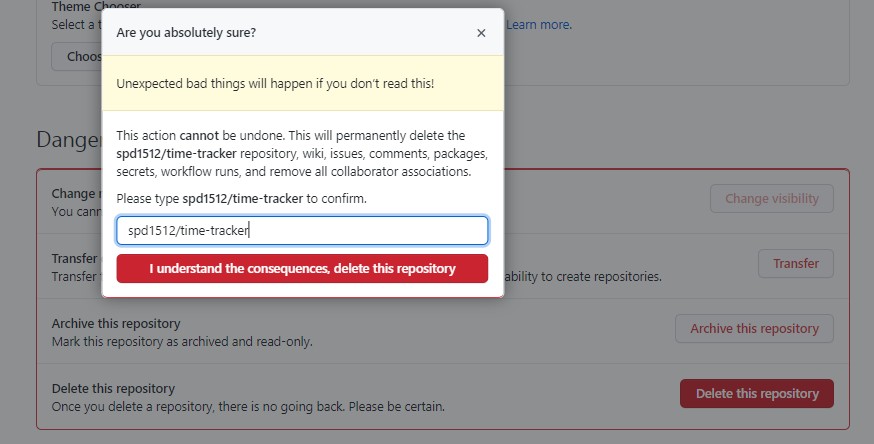
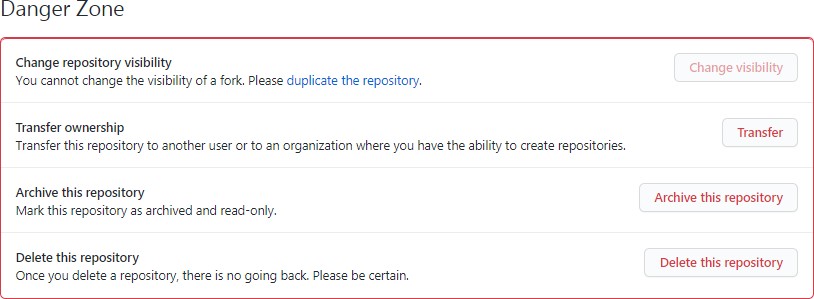
* By default, we can create public repository in Github. So we can copy the entire public repository of any other users in to own account using “FORK” Operation. Now fork the repository (Sharing with other users who wants to contribute).
* Login with another account Copy and Paste URL of repository then just click onto clone to others account. Suppose we want to fork public repository “timetracker”. So search for “timetracker” github repository on google and once its opened clicked on “Fork button” from the top of the github web page as shown below.



* After fork it will be added in your local repository.

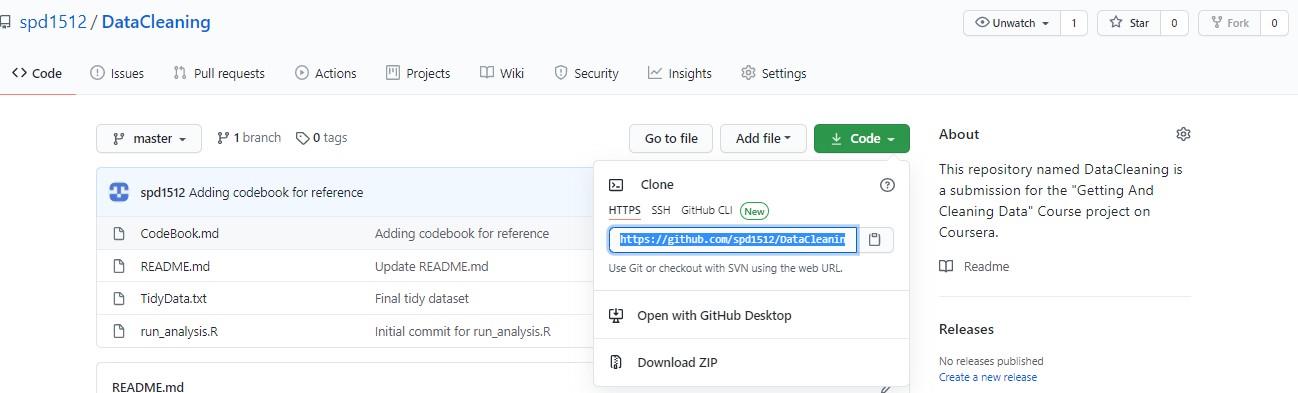


* To delete the repository, open the desired repository you want to delete and go to the settings option. There you will see delete repository button to delete it.



**Git Clone**

* Now, if you want to download a repository in local machine, then git clone command is used followed by path to repository.
* In GitHub the path of repository can be known through clone or download button and it can be downloaded using git clone command as shown below.



Text

Description automatically generated

Branching

Feature branch

Commit 3

Commit 2

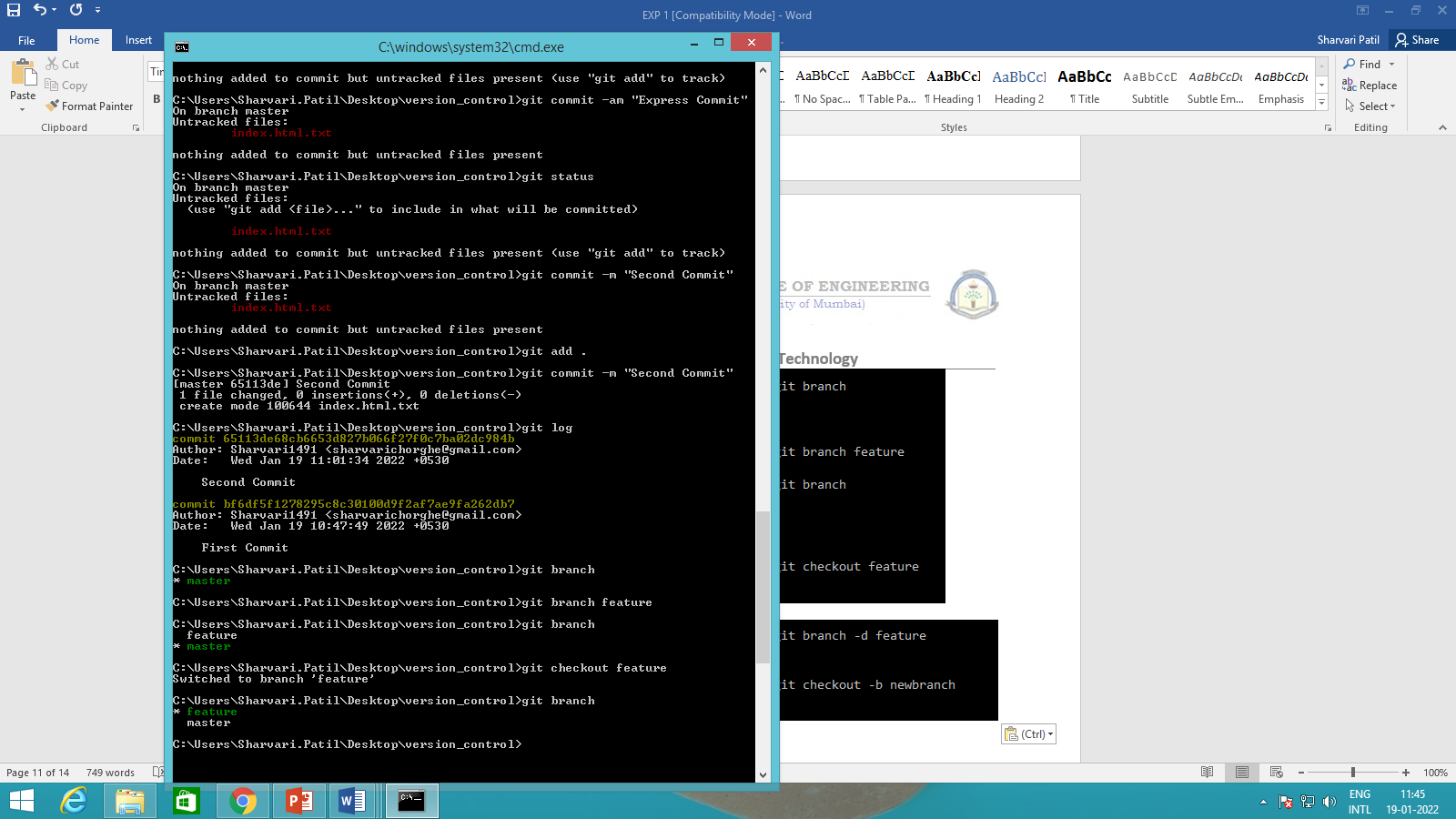
Commit 1

Merge Commit

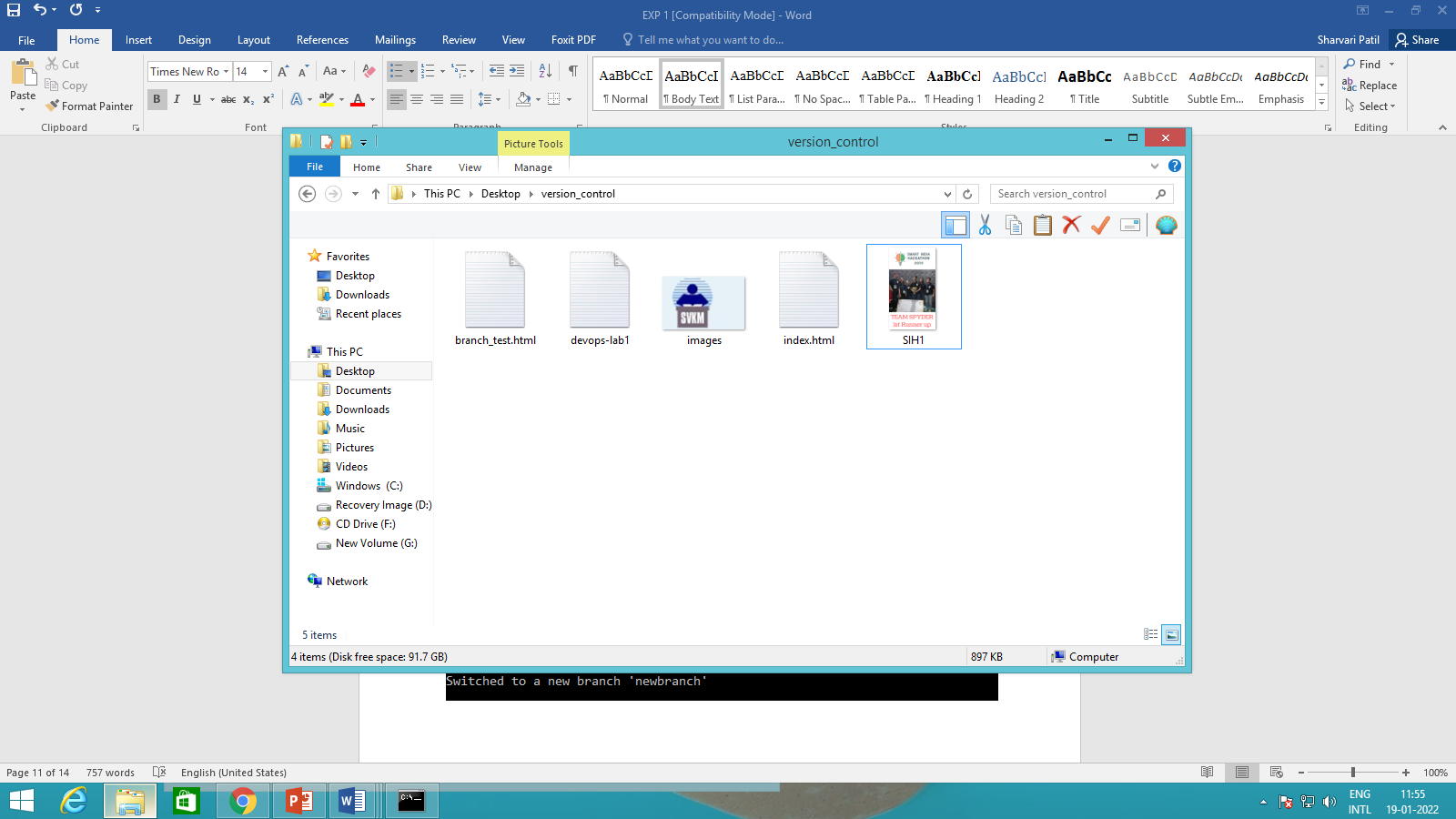
Commit 2

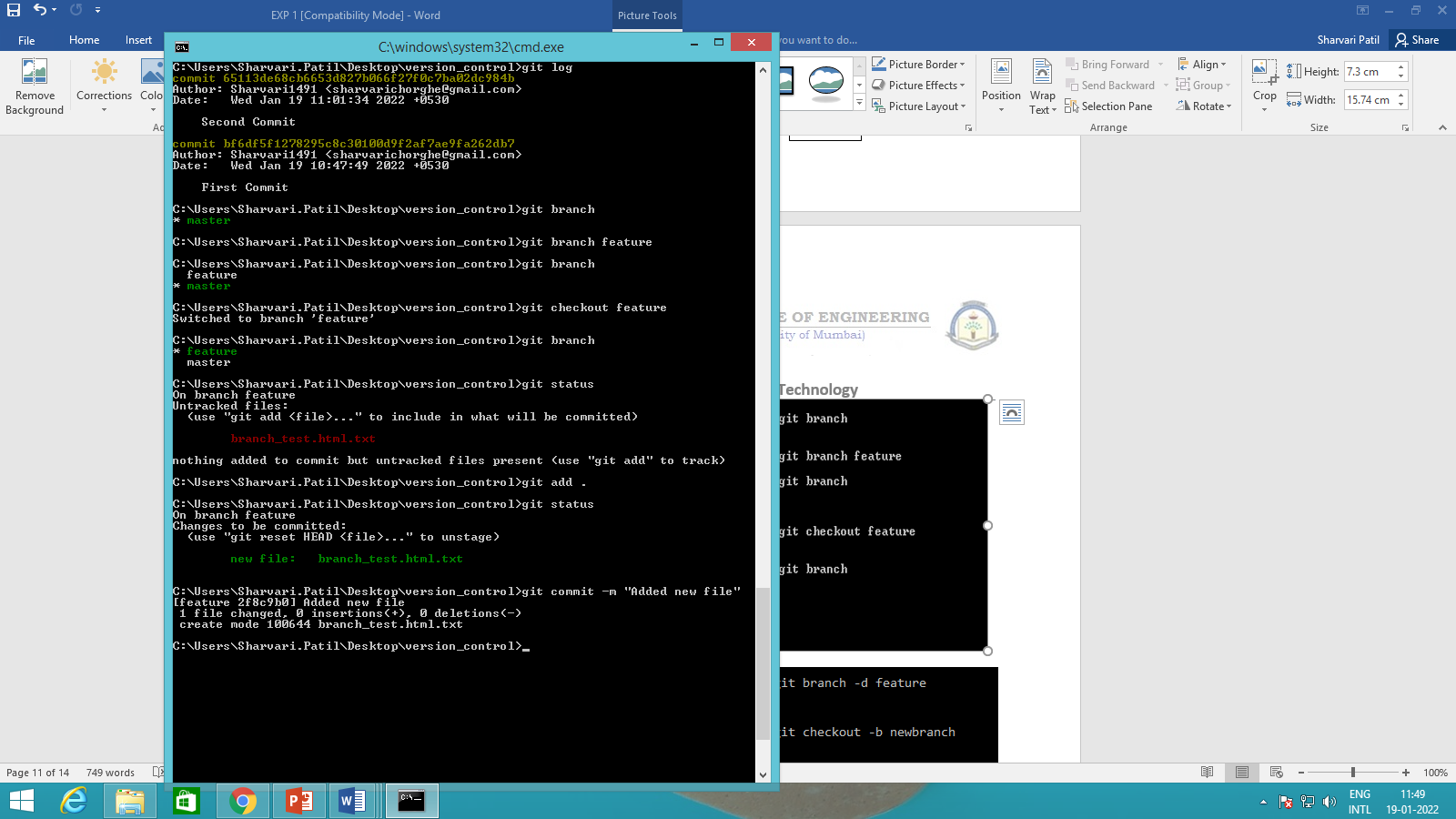
Commit 1

Master branch



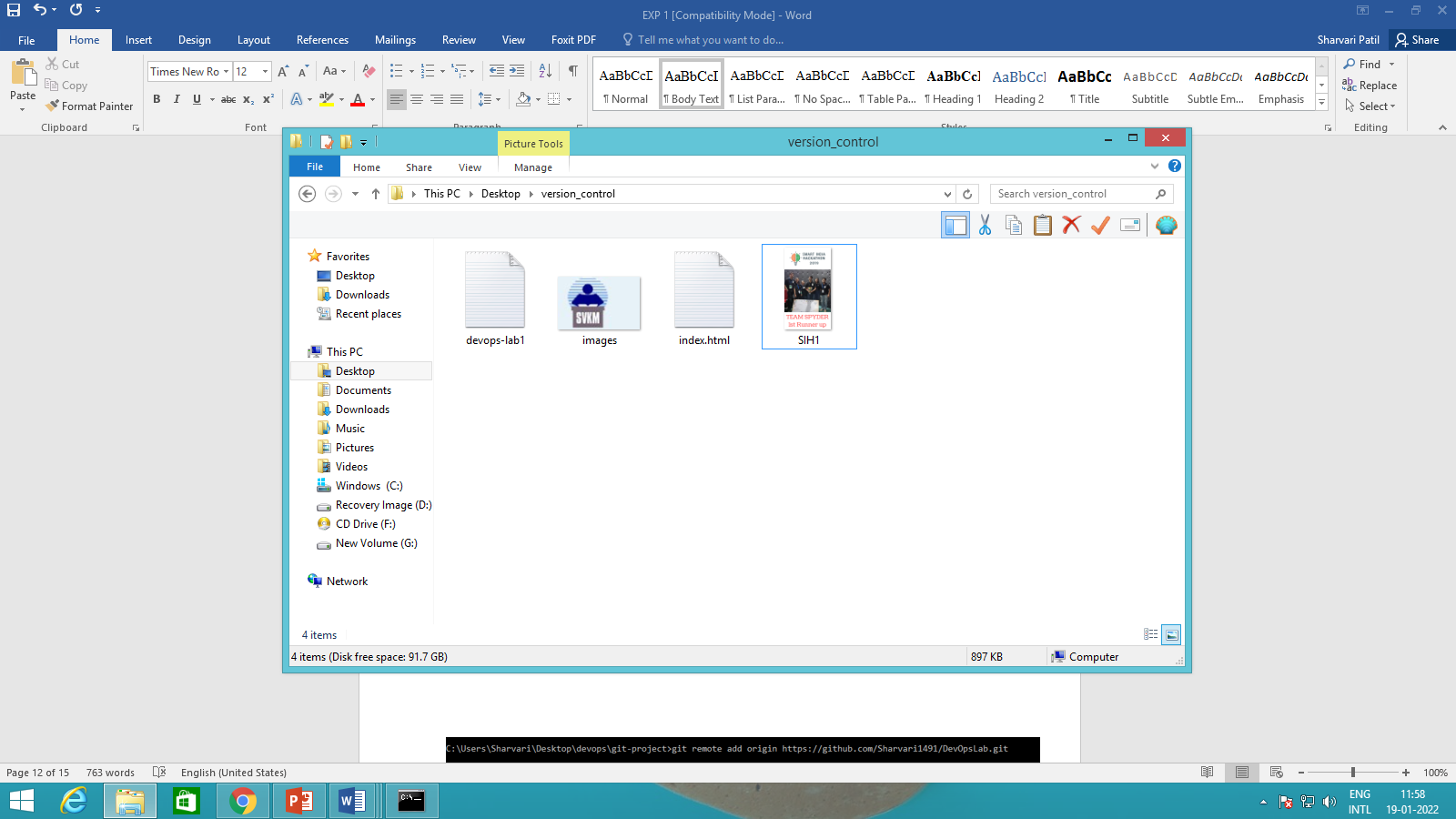
* **Add a new file in the folder branch\_test.html**





**Checkout to master branch and check**



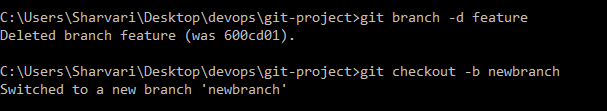


**To delete a branch**

**git branch –d feature**

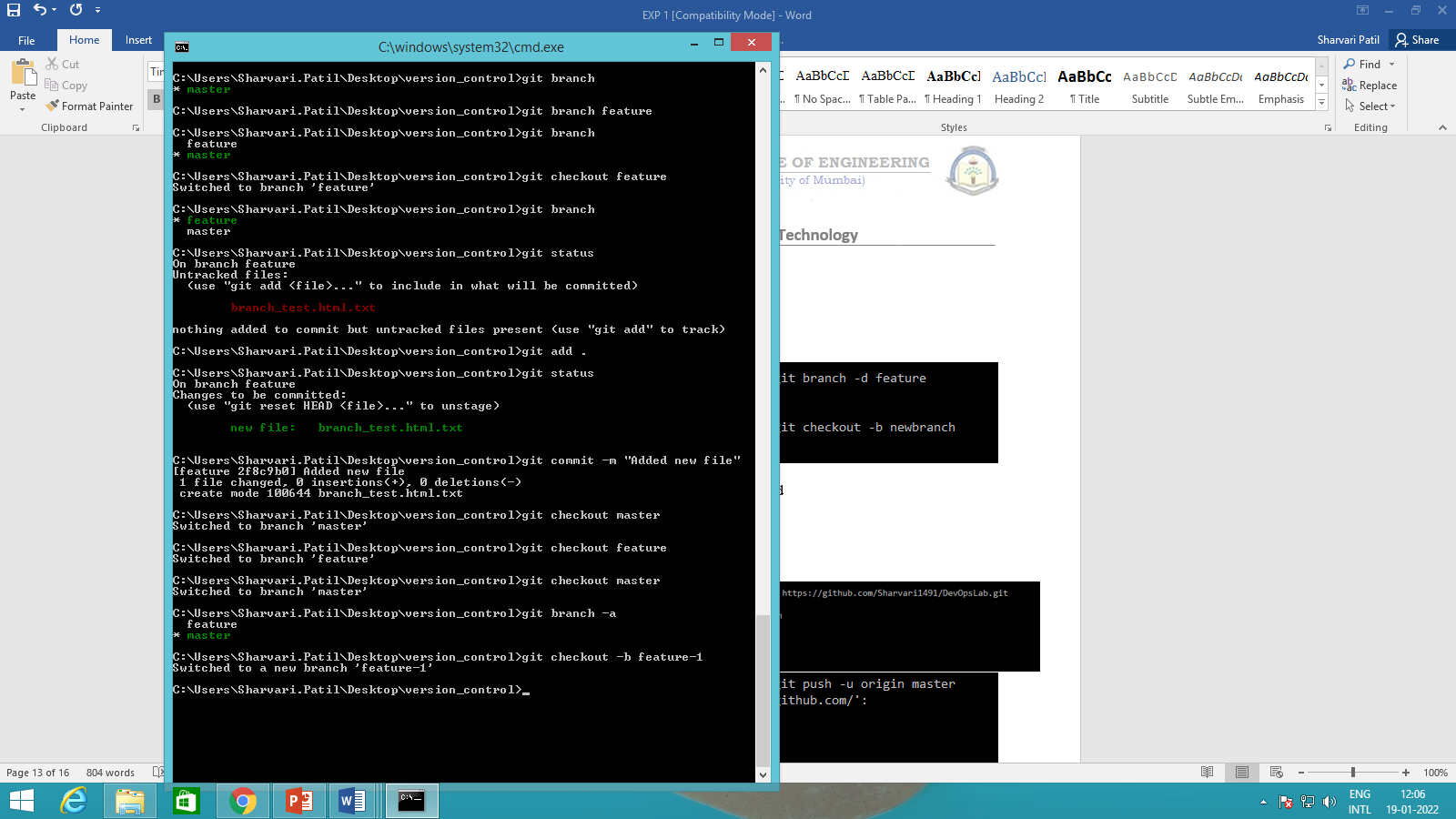
**it will throw an error if the branch is not full merged.**

**Use git branch –D feature**



To create a new branch and switch to it use the command

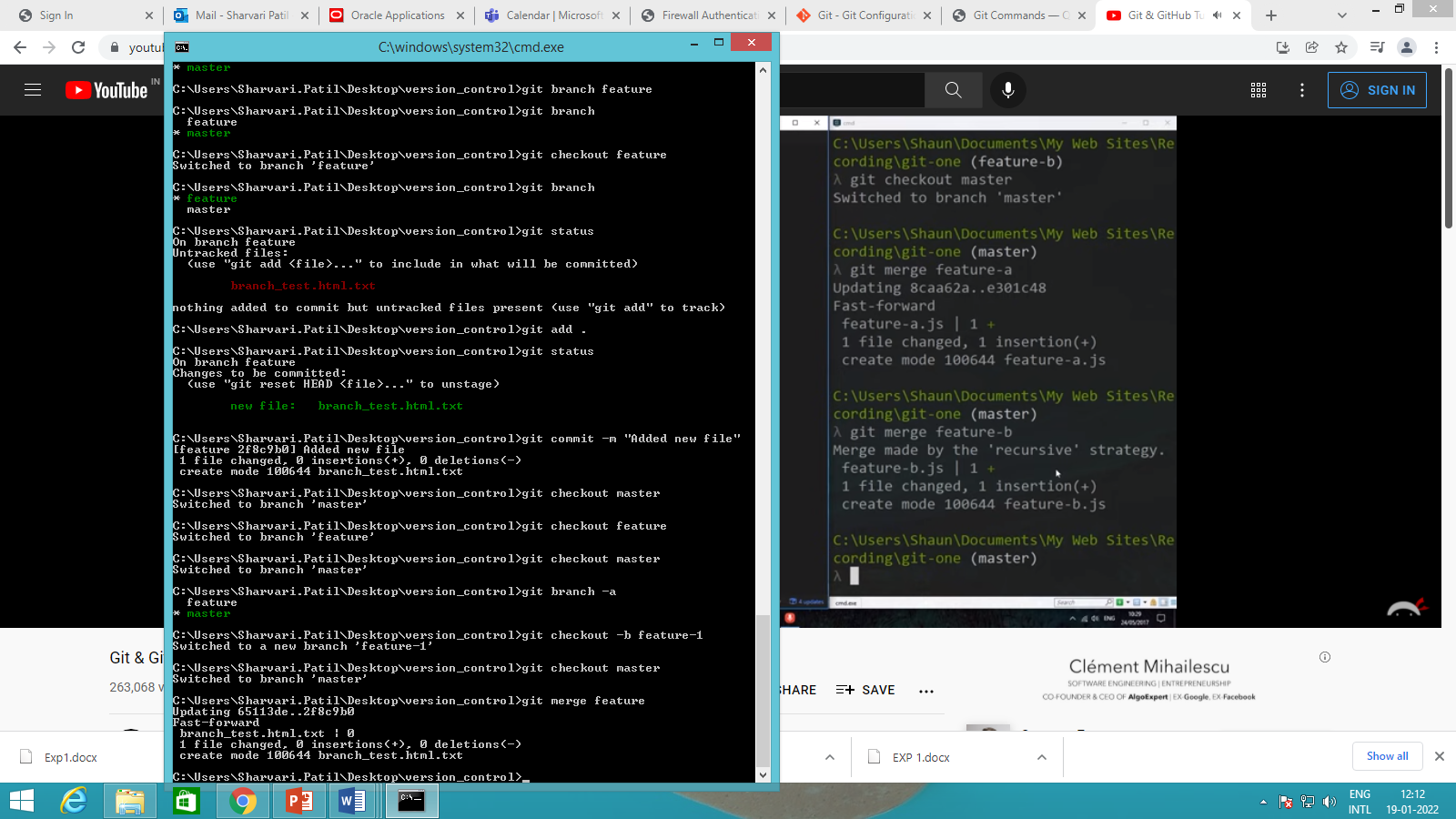
**$git checkout -b “<new-branch-name>”**



To merge the branch into master first switch to master branch

git checkout master

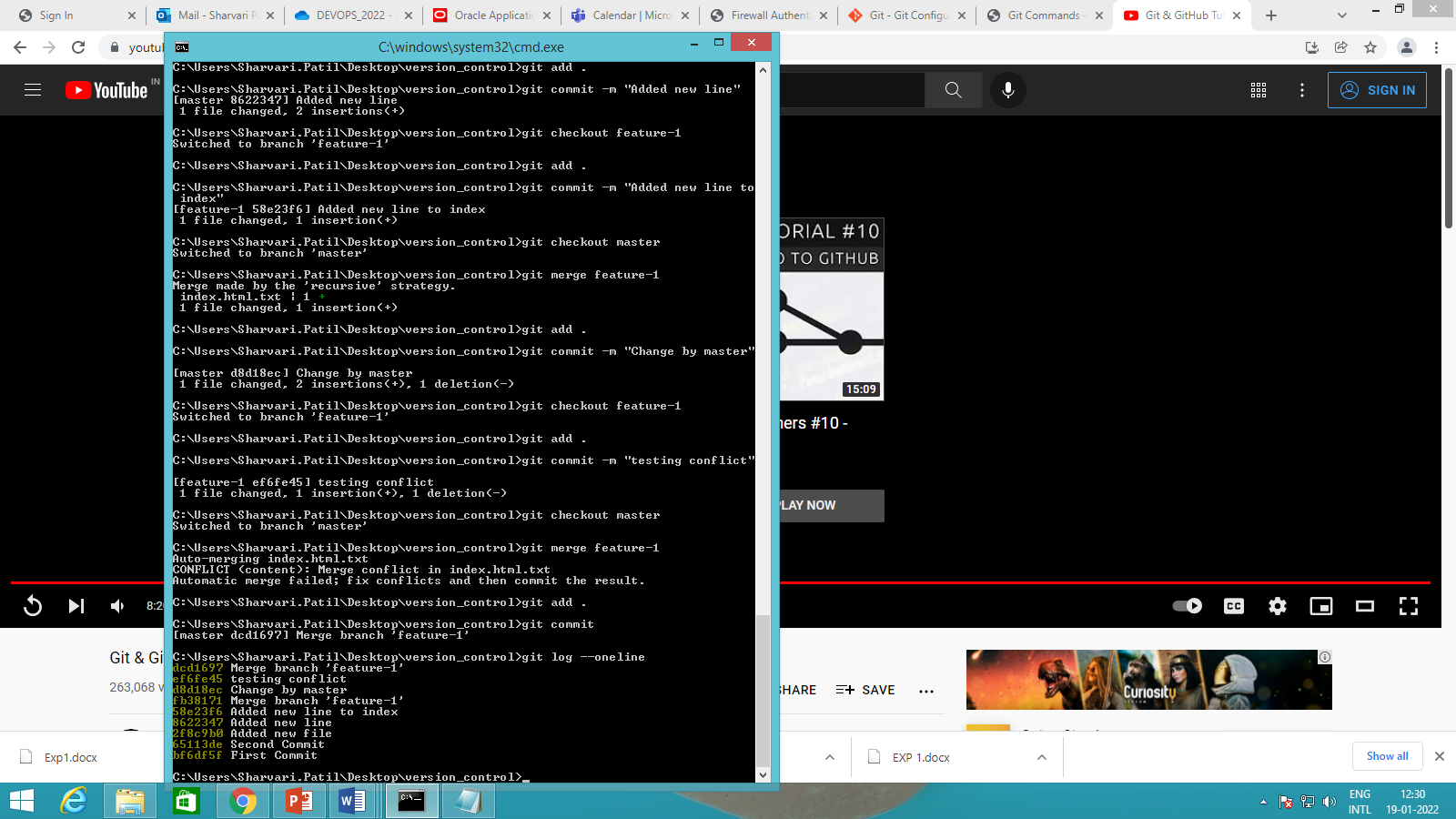
git merge <feature name>



**Exercise: Make changes to a file in master branch and perform commit**

**Update the same file in feature branch and commit.**

**Switch to master branch and then perform merge operation.**



# Pull and Push Processes

* The pull command used to fetch the repository from github to local while push is used to commit files from local repository to Github.
* **Push:** Push changes to Web repository
* **Pull:** Pull changes to Local repository
* The following commands are used for pull and push repositories:

1. **Push Command**

**$ git remote add origin https://github.com/Sharvari1491/DevOpsLab.git**

# $ git remote show origin

* If you add remote again then will show you fatal error.

$ git remote add origin <https://github.com/spd1512/My_repository.git> fatal: remote originalready exists.

* Now, to push the local repository to remote github following command is used

# $ git push -u origin master

* So, to delete origin rm origin command is used

# $ git remote rm origin

* Now you can check the github for updated contents.

Text

Description automatically generated

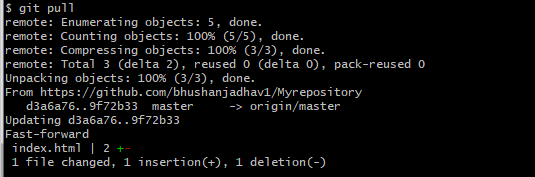
Text

Description automatically generated  
Text

Description automatically generated

**Pull Command**

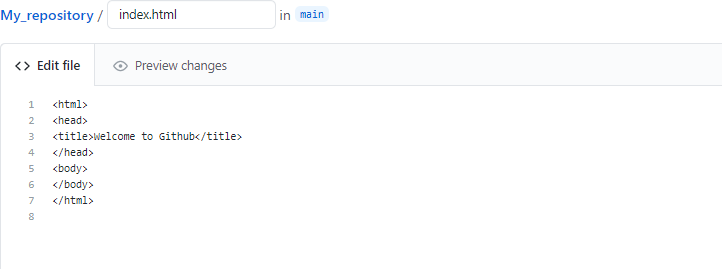
* Pull command is used to download the remote updated repository into local one.
* The command for download is **$ git pull**



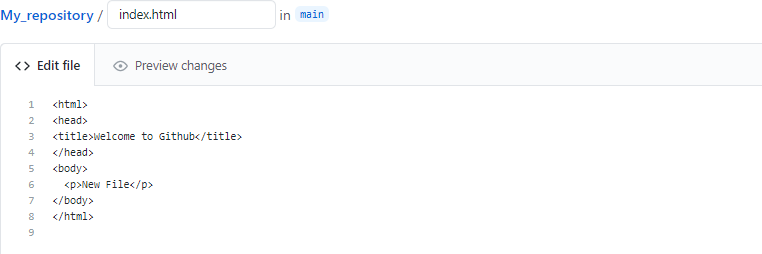
* Now you can see the changes in local repository using git log.

1. **Fetch Command**

* Suppose you have a file in github and you have changes that.



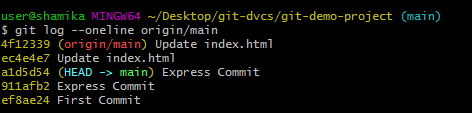
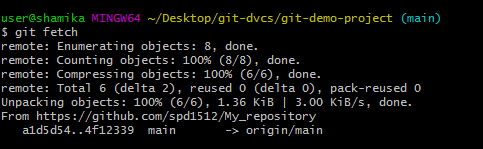
# Original File



**Changed File**

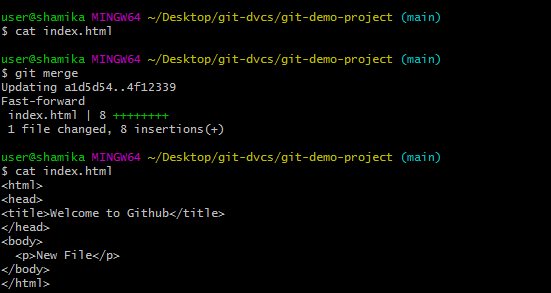
* Now we use fetch command to fetch the changes, which will show you both the files like original and changed in local repository.

# $ git fetch



* Here fetch will not show you like updated changes file as like push. So use merge command to merge the changes so use following command for merge.

# $ git merge origin/master



Text

Description automatically generated

# Conclusion:

In this experiment, we successfully learnt about version control systems, installed Git and performed basic operations of Git.

Git provides a way of keeping track of past versions of software and papers, making collaboration between various authors easy, and provides backup for your software. It has proven very useful to the open-source community and in academia as well. Git also has several other commands that can be useful to perform less standard actions such as remove sensitive data, clean up past history to save space, and so on. We have successfully understood the purpose of using a version control for collaboration. Application and usage of GIT Push and Pull commands was successfully executed.