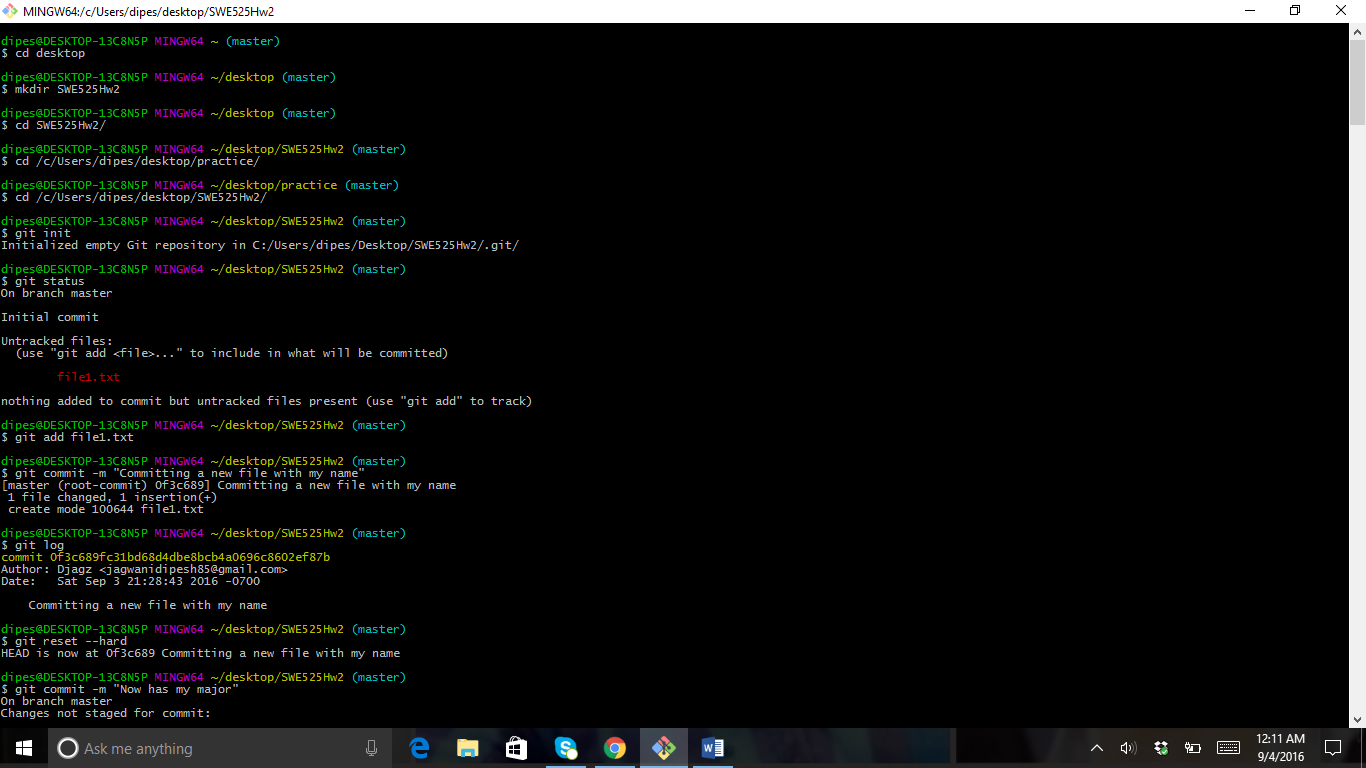
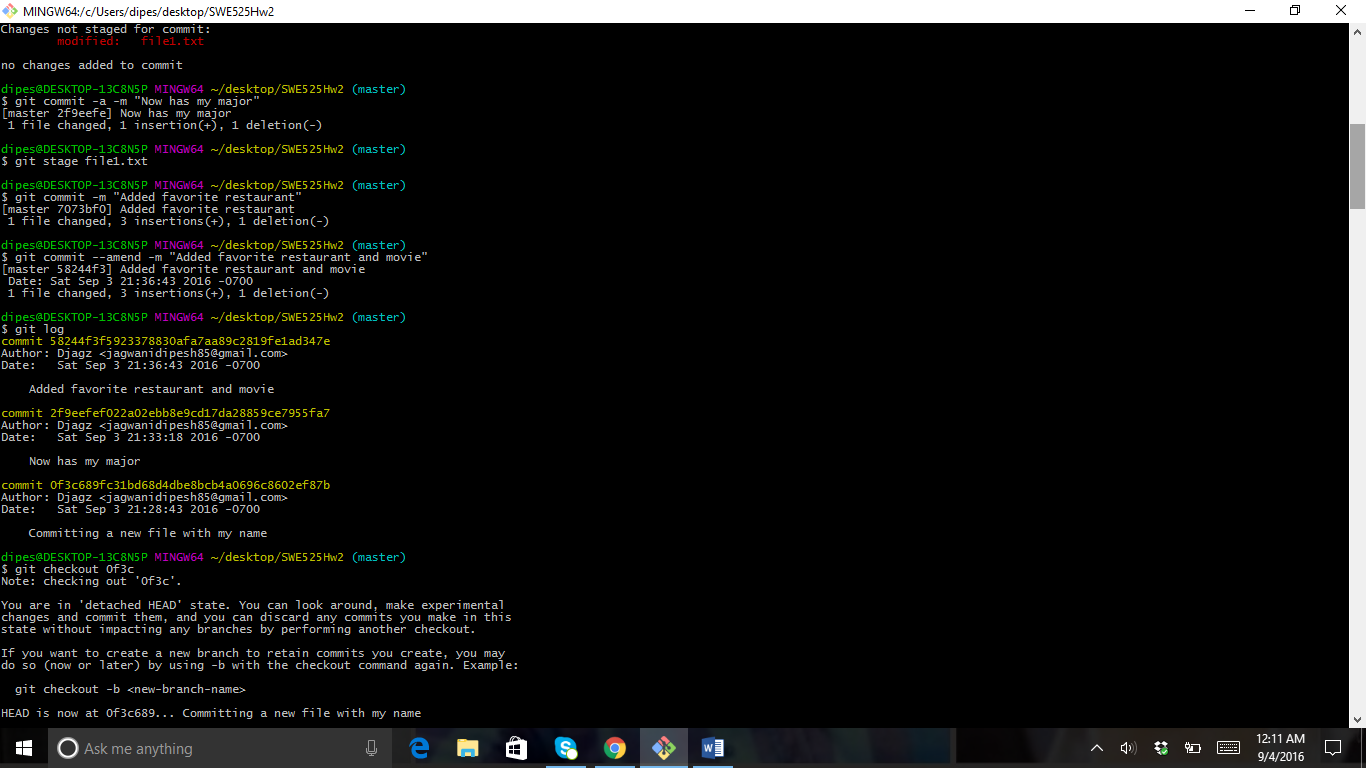
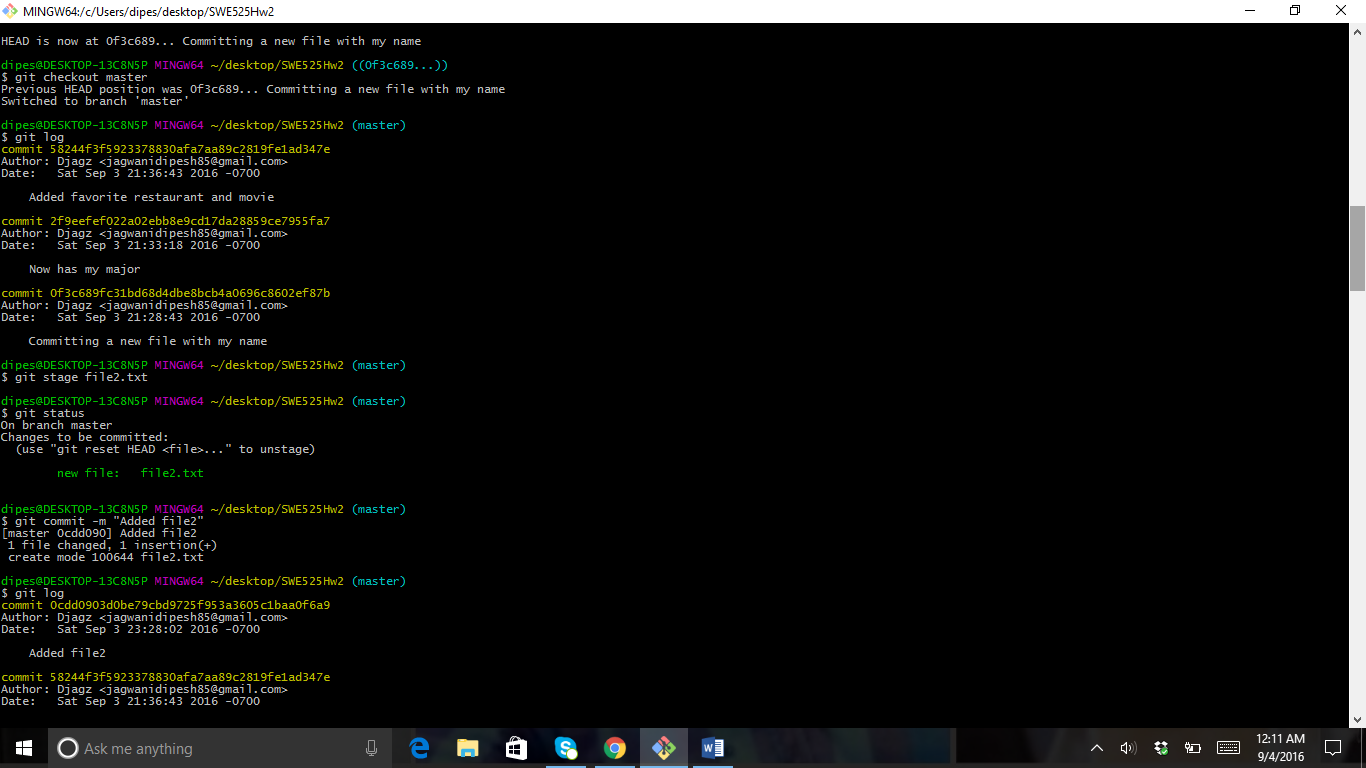
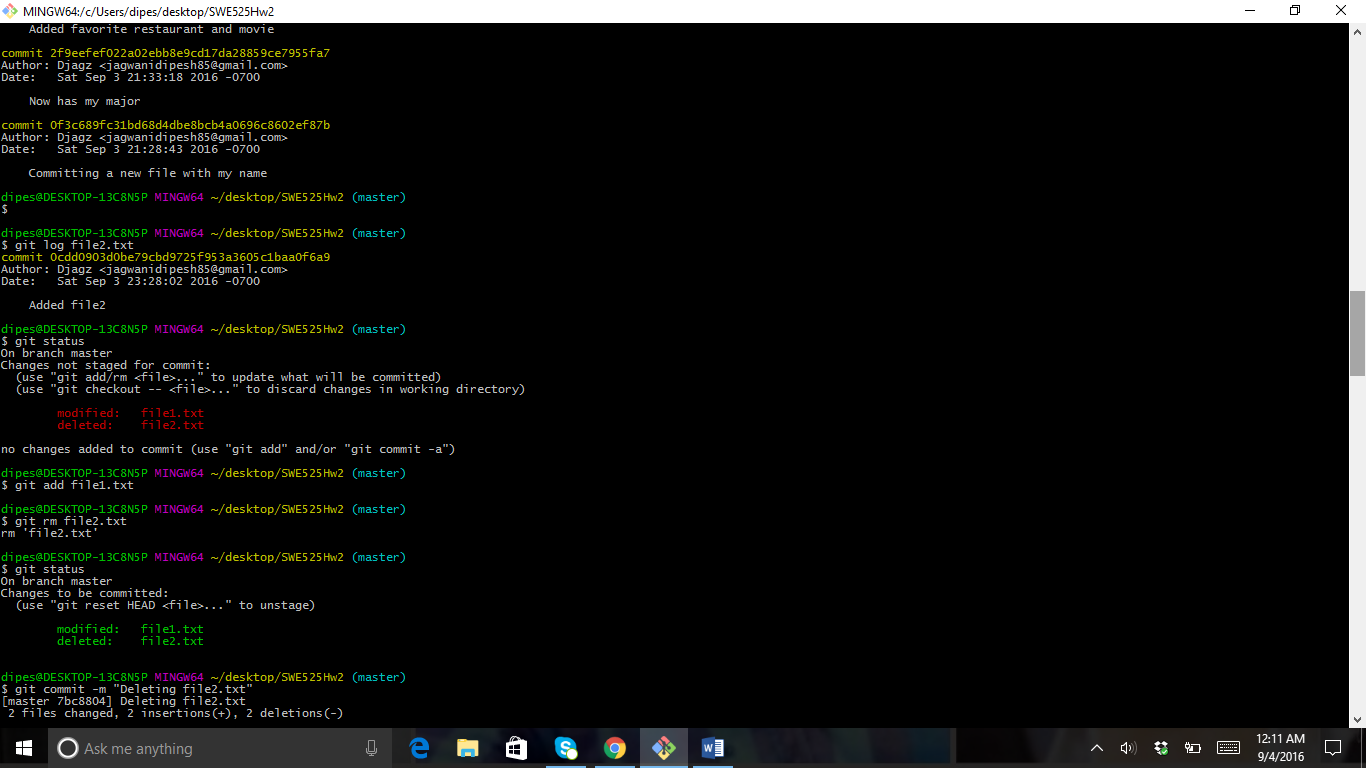
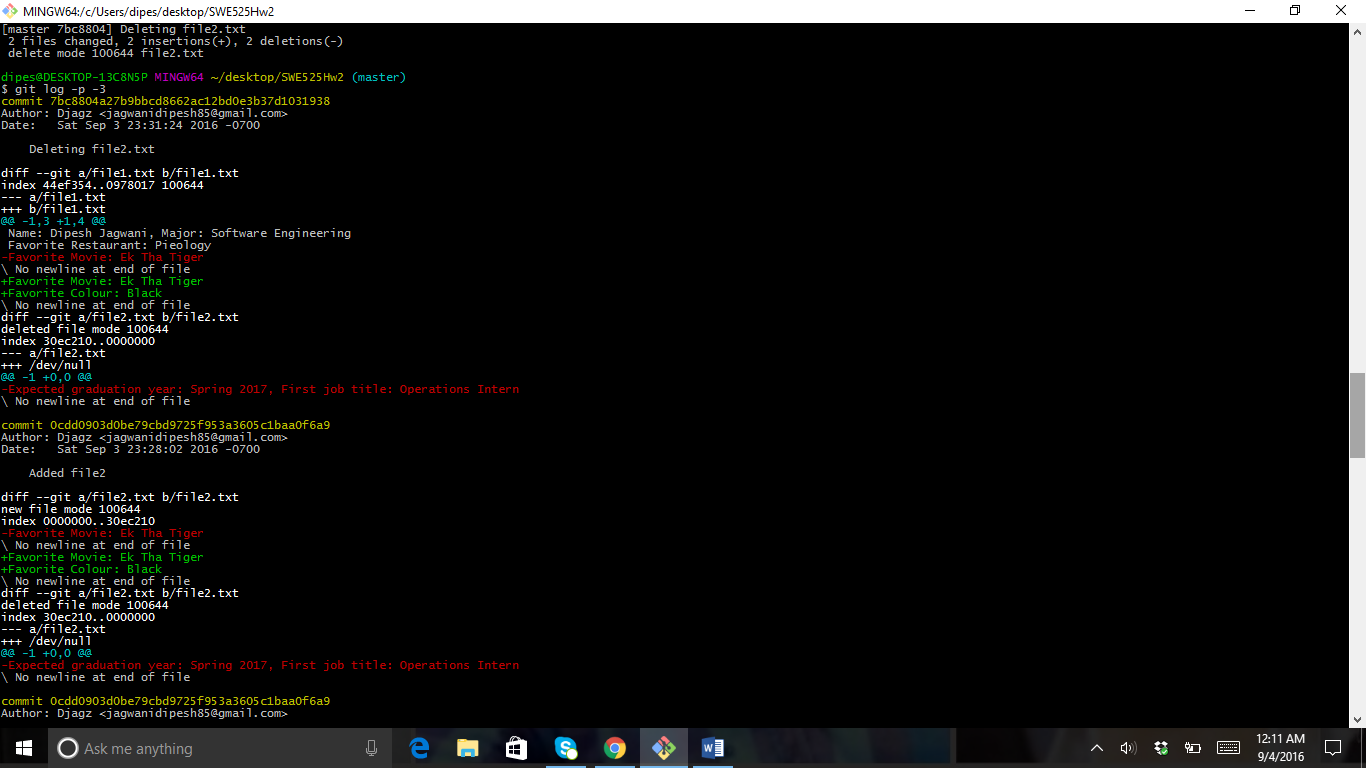
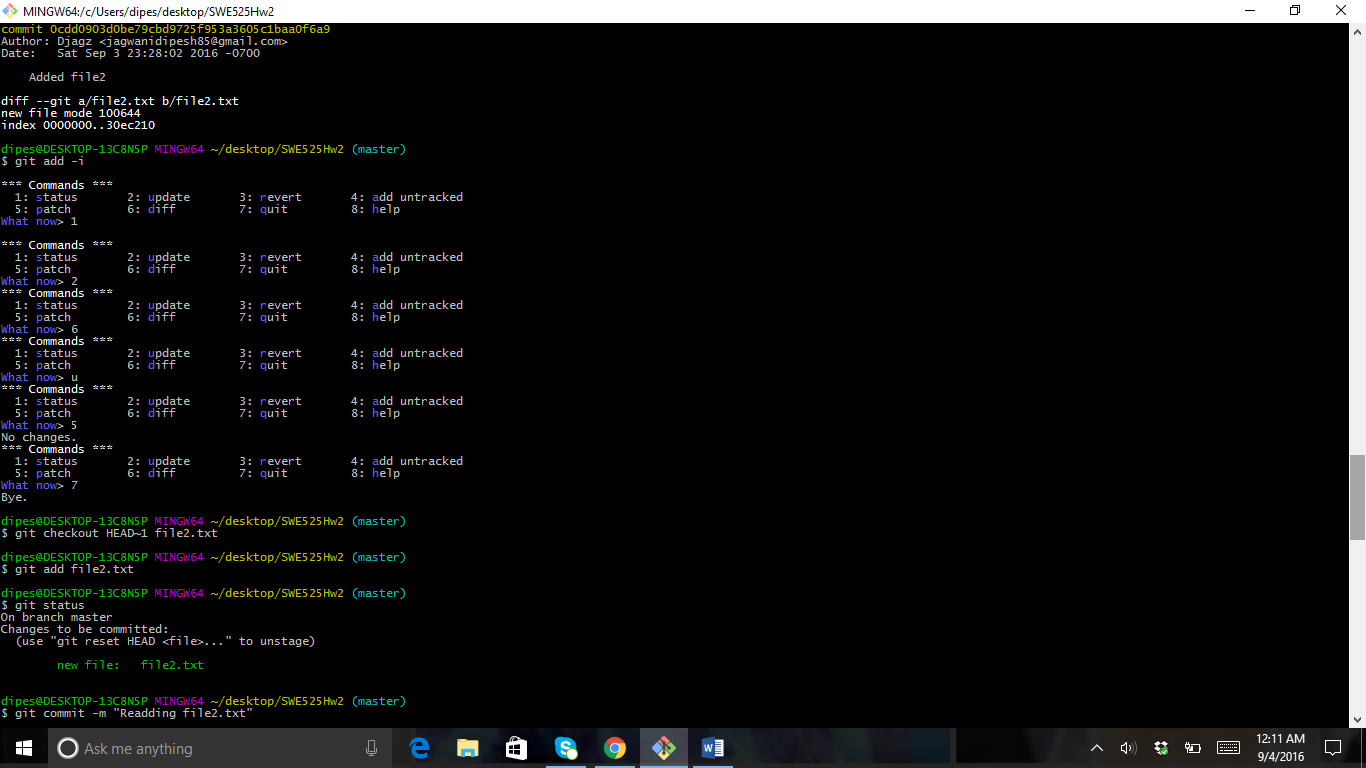
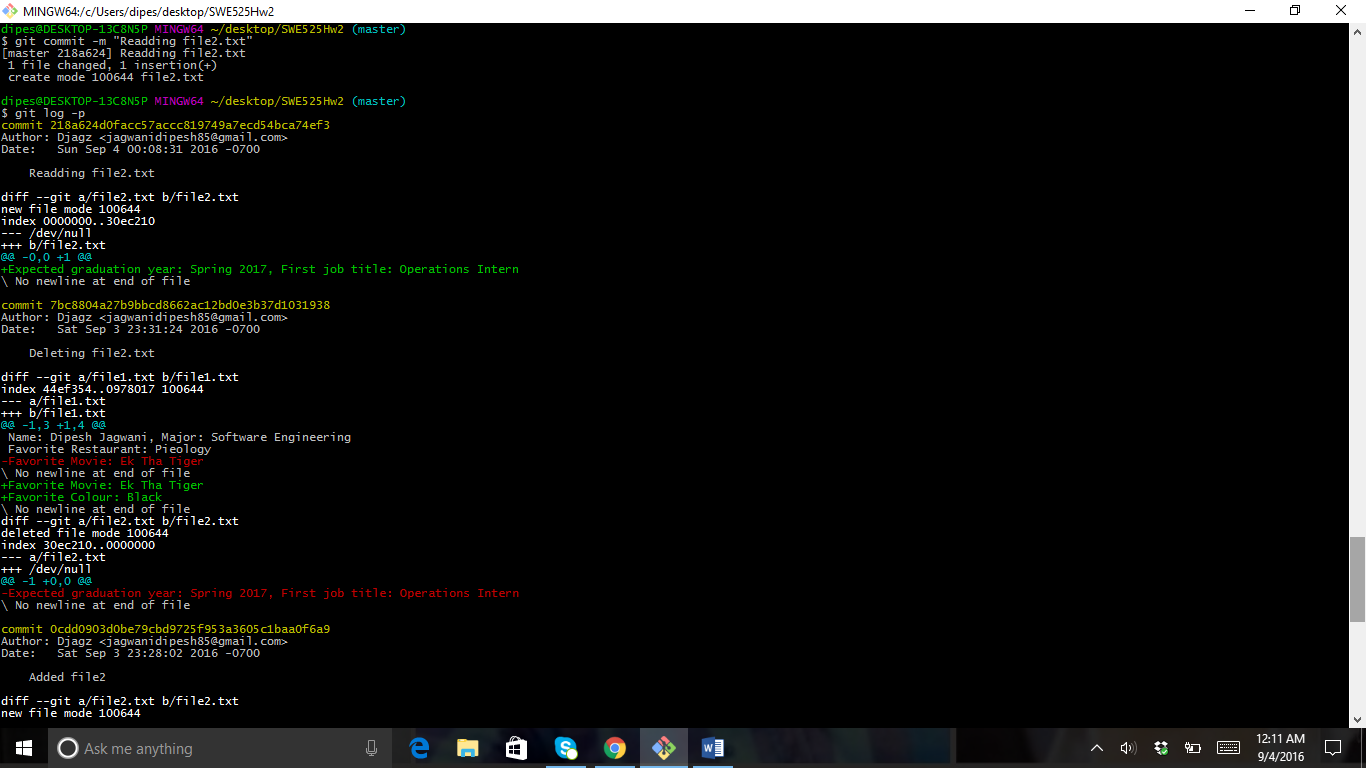
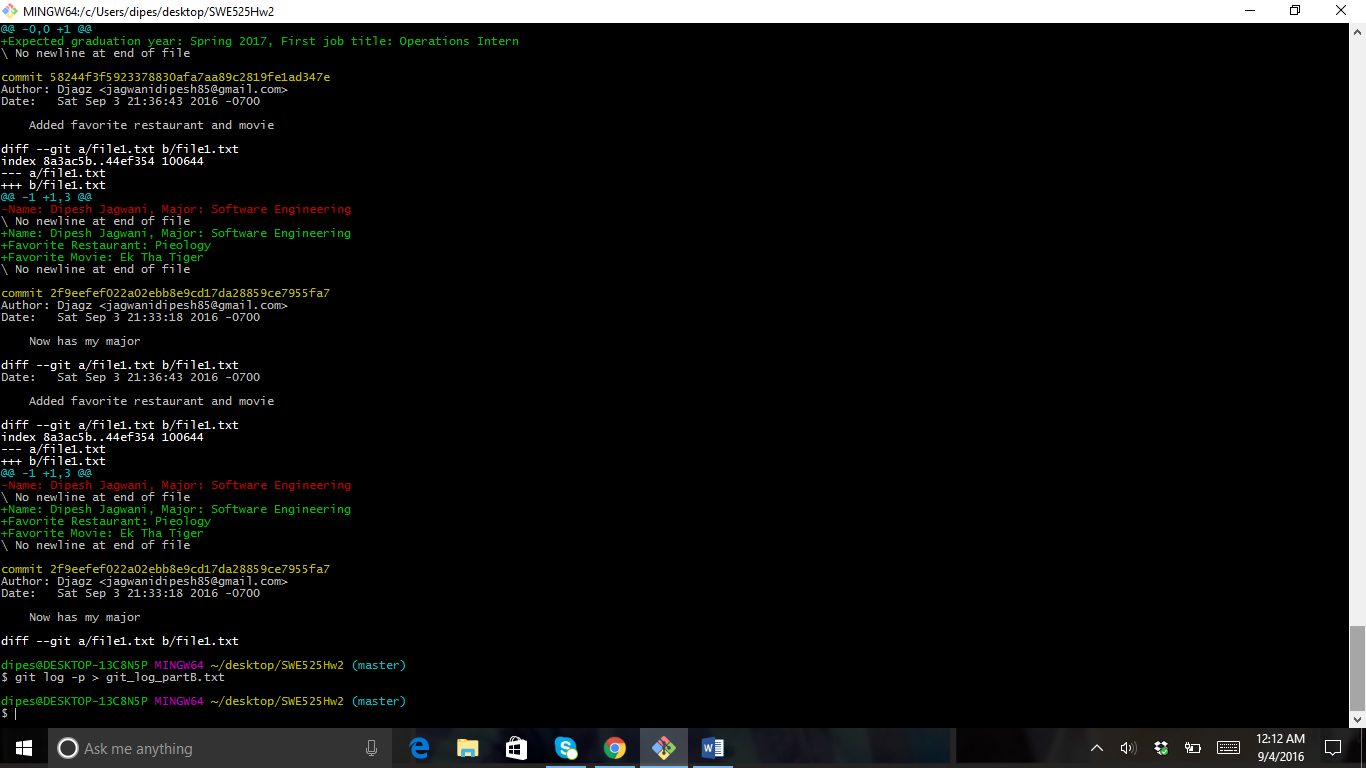
**SWE 525 Version Control Git Homework 2 08/23/2016**

**Name: Dipeshkumar Jagwani Student Id: 90443**

1. Part-A



1. Part B

All the commands in Part-B have been given, so I have done as per directed and I didn’t attach the screenshots because reddit had a lot of git log, so I have pushed my local repo to the github account and have submitted the log.txt file to the EMS.

1. Part C

To configure new git remote repo, I used “git remote add origin [url…]” and if I would have cloned directly from remote then git remote -v would have shown me links for my origin and that would be my remote url…

When I fetched with “git fetch [url…]” it configured my local with .git folder and other README.md files.

You can do a git fetch at any time to update your remote-tracking branches under refs/remotes/<remote>/. This operation never changes any of your own local branches under refs/heads

A git pull is what you would do to bring a local branch up-to-date with its remote version, while also updating your other remote-tracking branches.

When you use pull, Git tries to automatically do your work for you. **It is context sensitive**, so Git will merge any pulled commits into the branch you are currently working in. pull**automatically merges the commits without letting you review them first**. If you don’t closely manage your branches, you may run into frequent conflicts.

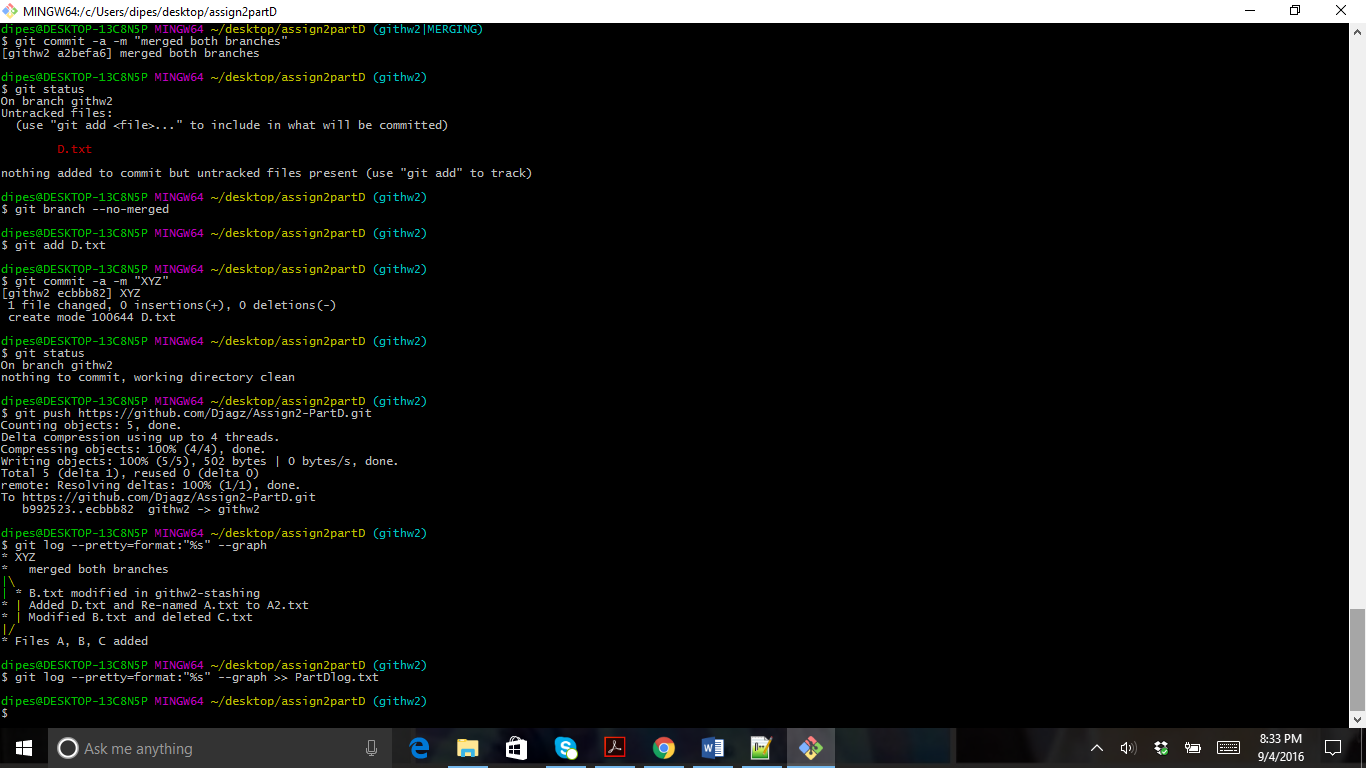
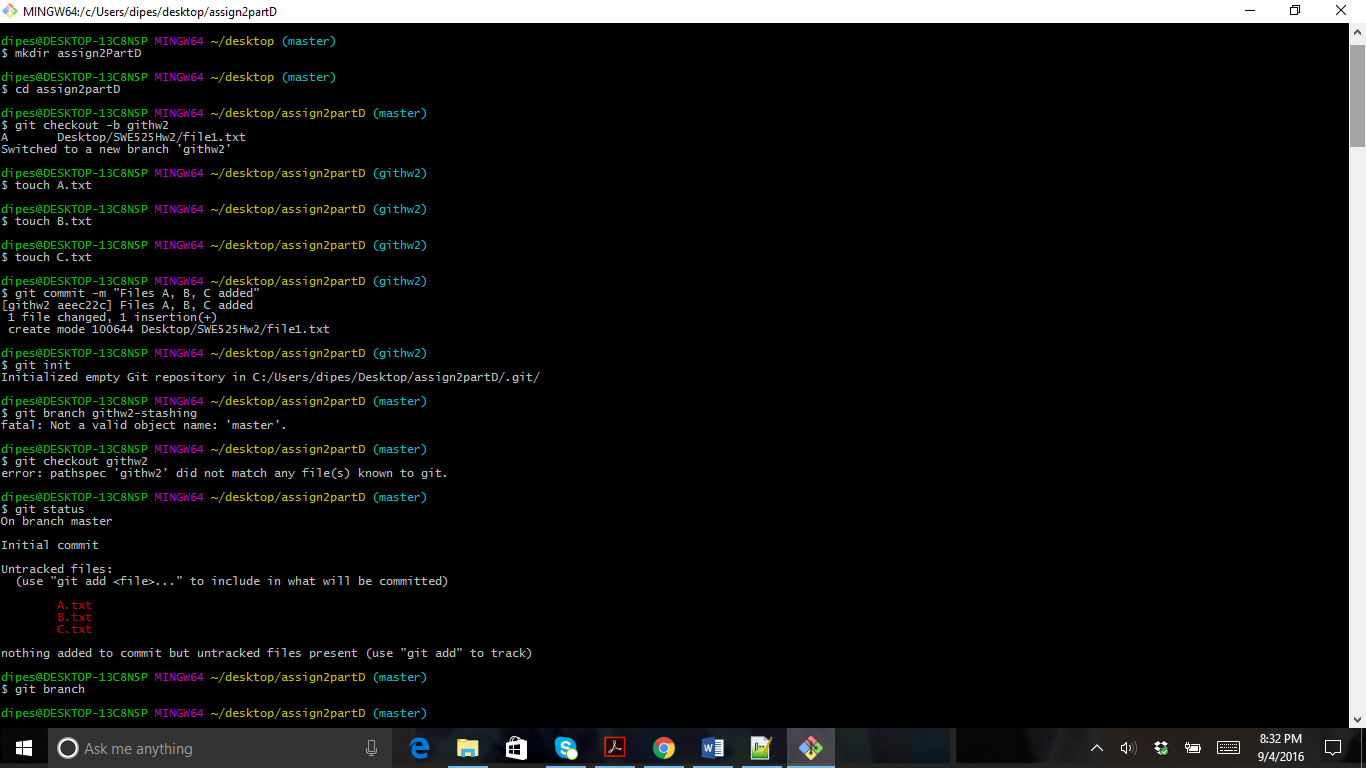
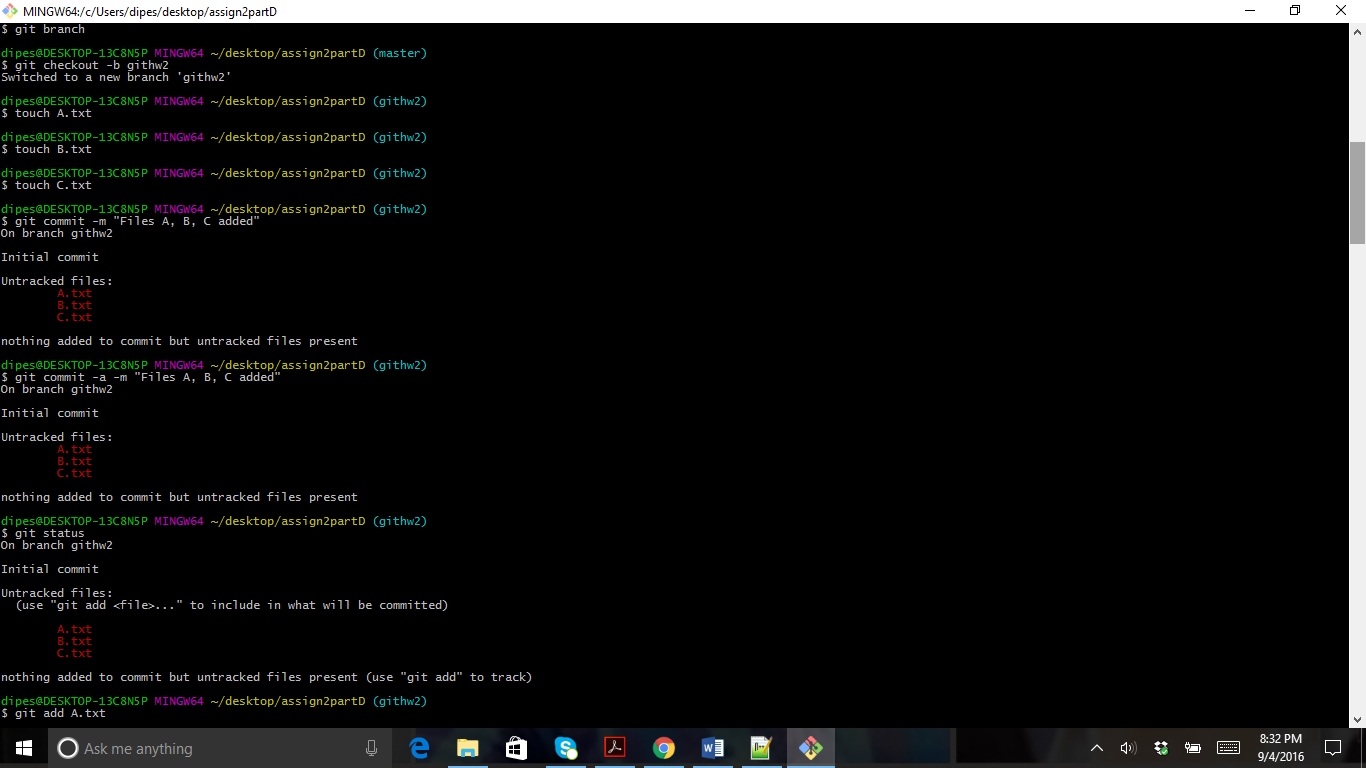
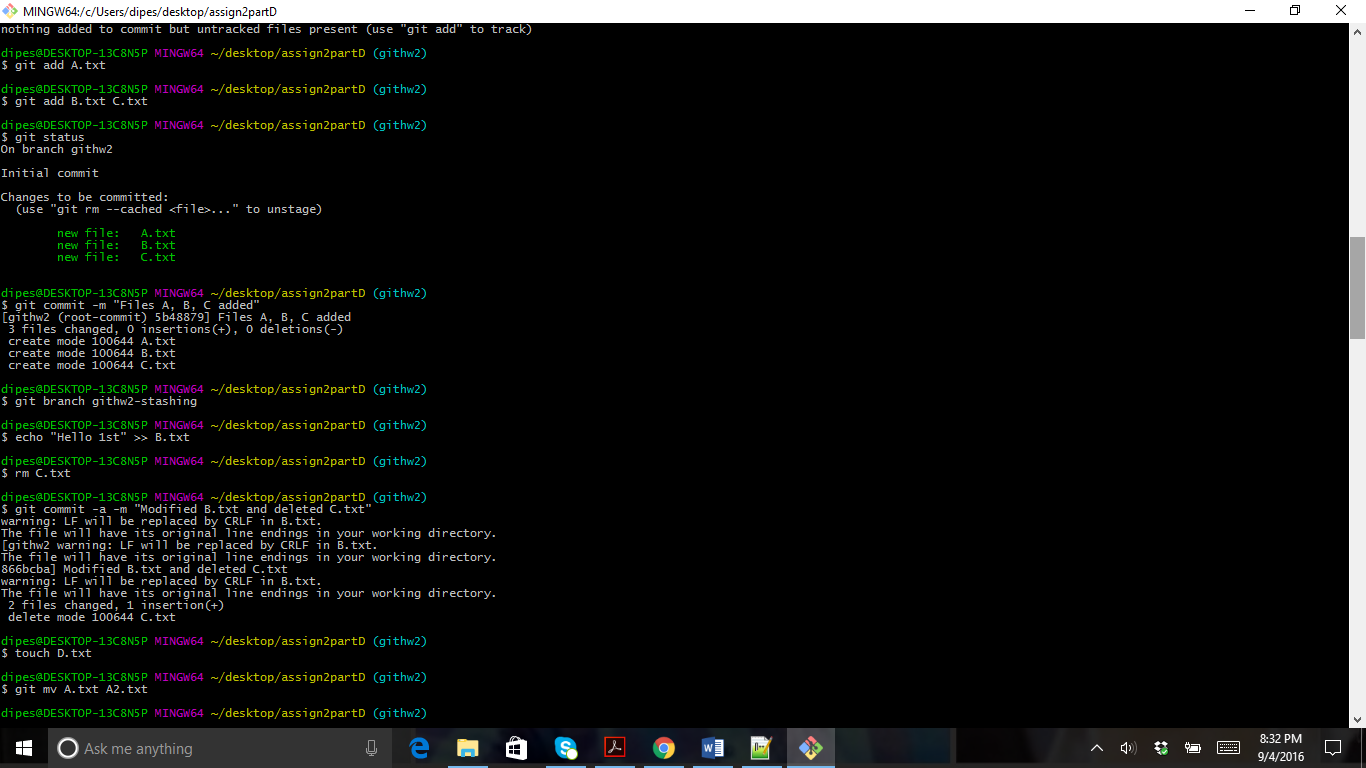
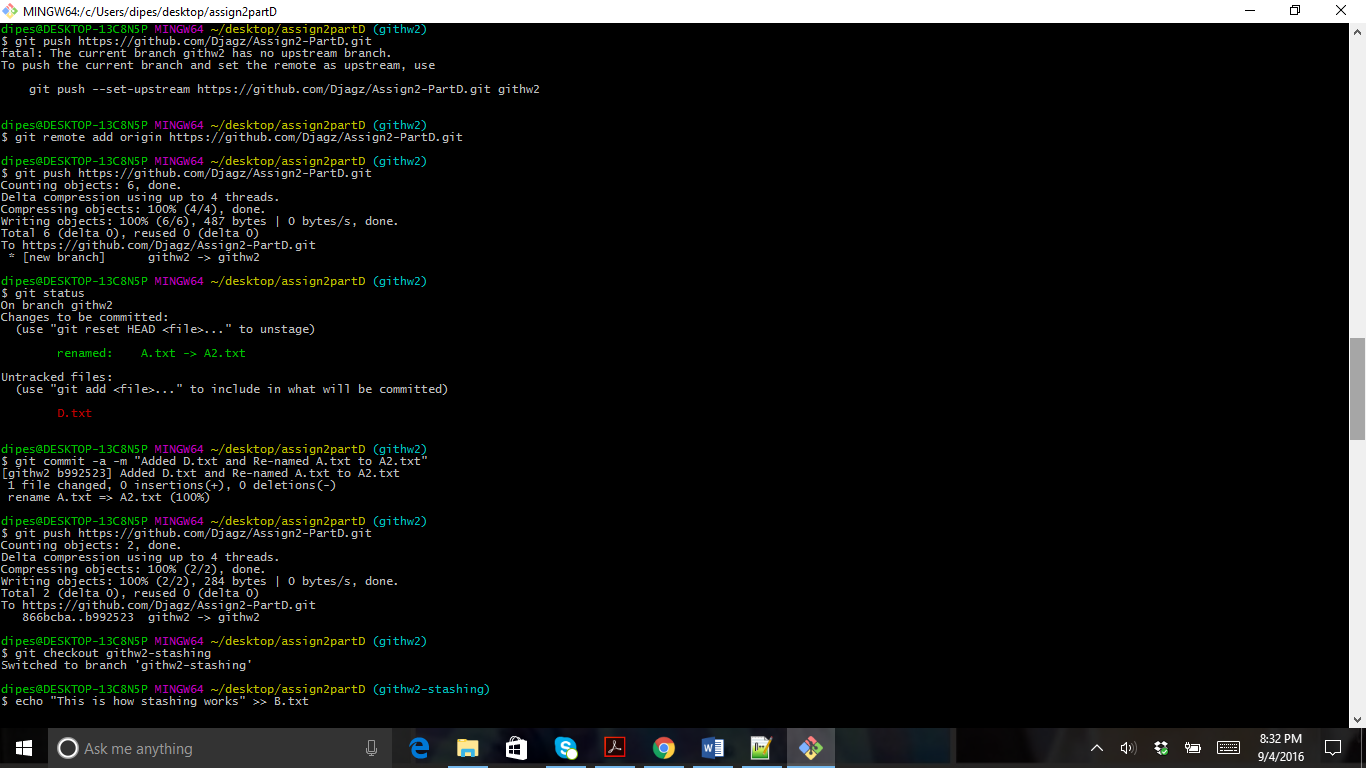
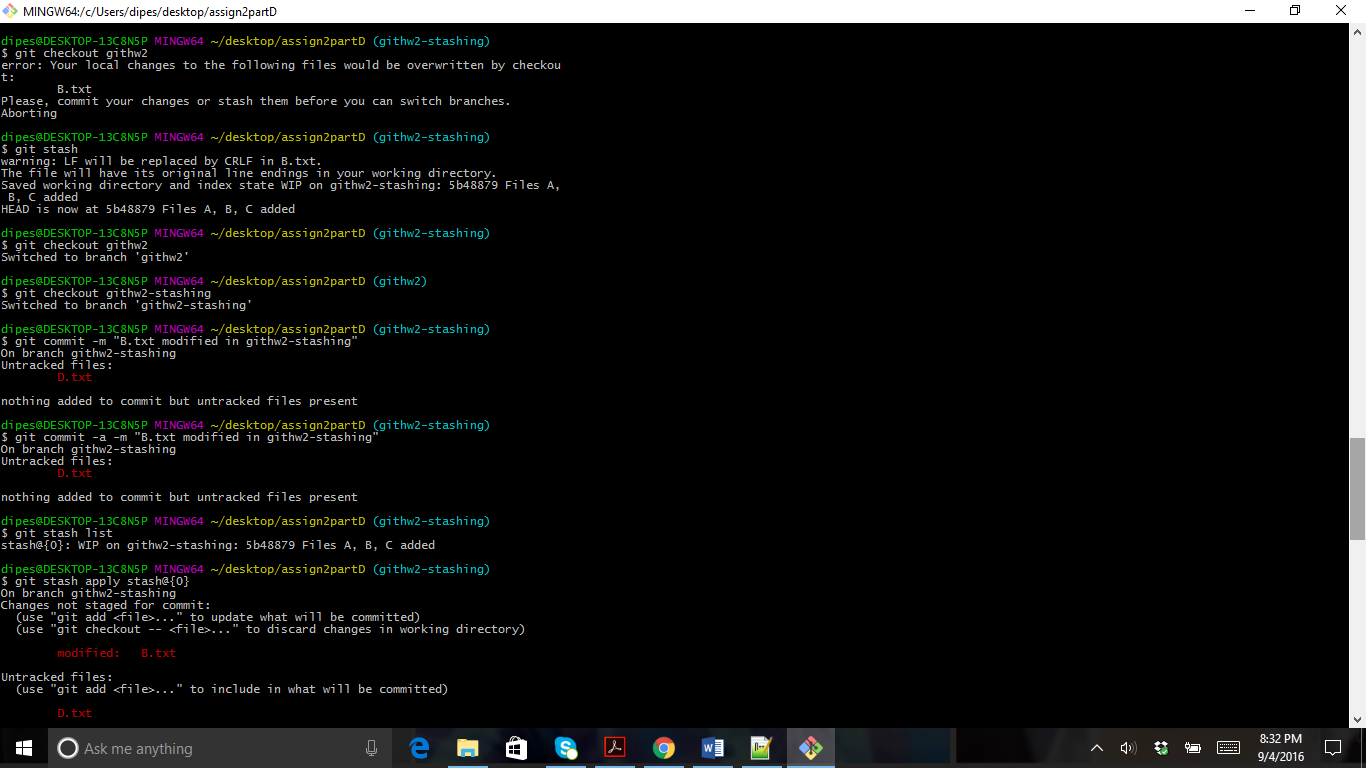
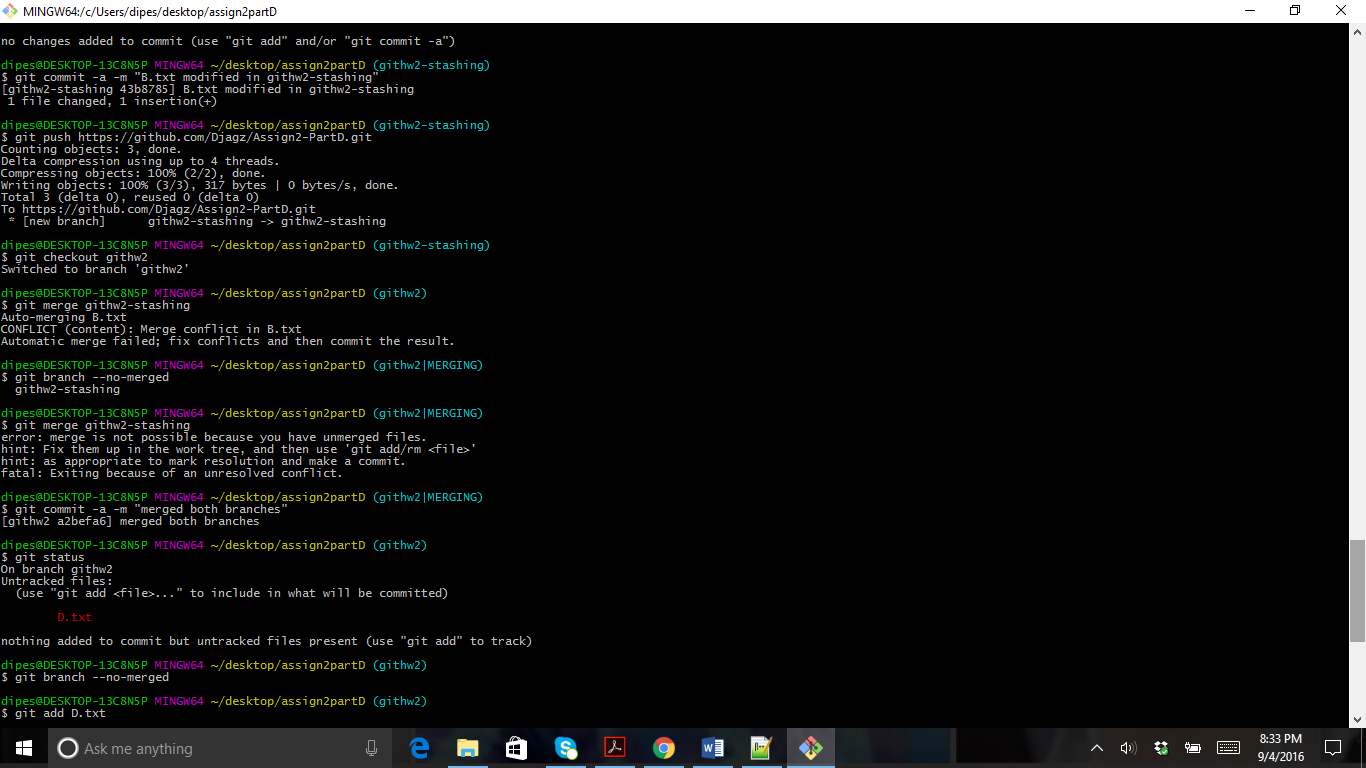
When you fetch, Git gathers any commits from the target branch that do not exist in your current branch and **stores them in your local repository**. However, **it does not merge them with your current branch**. This is particularly useful if you need to keep your repository up to date, but are working on something that might break if you update your files. To integrate the commits into your master branch, you use merge.

Homework is pushed to my git hub account in Part C as mentioned in the assignment.

1. Part D

Normally a stack is a heap but it has some rules with it like LIFO (Last in First out). That means the commits will come in from last and we can only access them from the top not from anywhere in between.

While in stash we can use any commit we want from between, that means we can apply stash with any commit at any time by using “git stash apply stash@{id}”



**Part E:** Why don't you think git deletes branches you rebase from? (To clarify, I mean if you do git rebase develop while on featureA, git won't delete the commits where featureA used to be.)

Git rebase only changes the base and nothing else. If I am on master and last commit is “C” and I do git rebase develop then the last commit of master that is “C” will have develop as the base and it will not delete that commit.

