**SE3512 Software Construction & Development**

**Assignment: 05**



**Submitted By:**

Muhammad Arslan (BSE191032)

**Submitted To:**

Sir Samir Obaid

Department of Computer Science

Capital University of Science & Technology, Islamabad

# Q no: 01

**What is GIT?**

GIT is a Distributed Version Control System that allows developers to collaborate and work on projects at the same time while keeping track of and preserving a comprehensive history of changes to their files.

The following are some of Git's features:

* Keeps track of the past
* Open source and free
* Encourages non-linear growth
* Creates backups
* Scalable
* Encourages teamwork
* Branching is easier
* Distributed development

# Q no: 02

**What are GIT workflows?**

Workflow refers to the process of regulating the events that convert a piece of work so that everyone on the team may utilize the correct tool at the right moment. Simply described, a workflow is a method for completing tasks. At its foundation, a workflow is change management. It is a sequence of actions you must execute in order to accomplish certain recurring business goals. Workflow comes in a variety of sizes and forms. The transformation material delivers services or processes information, and an effective workflow allows for the systematic organization of resources into processes.

The following are some popular Git workflows:

1. **Centralized workflow:**

The centralized workflow uses a single branch to accomplish the work of the project. Even though is very simple we still get benefits of Git. Your team members cannot work independently and each as a local copy of the project history.

Master

**Remote Repository**

Master

**Local Repository User B**

**Local Repository User A**

In this example, the remote repository contains two commits. User A clone, fetch or pull, when the remote repository only had commit A, they then created commit C. To add C on the remote repository the user will have to pull or fetch and merge before pushing commit C. User B currently has both commits from the remote repository and be able to push to remote repository with no problem.

The down side of Centralized workflow is that you are not taking advantage of feature related to branching such as pull request.

1. **Feature Branching:**

In a feature branch workflow, the work of the project is done in feature or topic branches the work is then merged in a longer running branch. The feature branching workflow uses a single remote repository team members create feature branches and can submit their work using pull request.

**Remote Repository**

Feature 1

Master

**Local Repository User B**

**Local Repository User A**

Feature 2

Feature 1

1. **Git-flow workflow:**

Master

Master

1. **Forking workflow:**
2. **Forking workflow:**

The forking workflow involves multiple remote repositories. One of the repositories is consider upstream from the other. The upstream repository is considered the source of truth for the project. Work is usually transferred from the remote repository to upstream repository via a pull request.

One advantages of this workflow is that user of Forking repository does not mean to have right access to upstream repository. This is because a user on upstream repository merges the pull request because of this the Forking workflow is very common in open source project. Forking a repository is a great way to work on features branch without sharing a branch this provides a remote backup of your work and allows you to safely rebase your local branch.

The downside of this approach is the two remote repositories can become out of sync. It’s the responsibility of the forked repository to keep up to date with the upstream repository.

**Upstream Remote Repository**

**Forked Remote Repository**

Feature 1

Master

Master

**Local Repository User A**

Feature 1

Master

In this example we have the upstream repository on the right and the forked repository is on the left. You can see the forked repository created a feature one branch before commit C was added to the upstream repository. These remote repositories are now not synchronized. User A of the forked repository would have to synchronize repository before they could submit a pull request with a feature one branch this would involve merging or rebasing before making the pull request.

1. **Git-flow workflow:**

Git-flow is a workflow that allows save continues releases of the projects. It allows work to continue even through releases and half exist. This commit graph is an example in sprit of Git-flow workflow. The general idea here is used in a specific ways that depend on the team and the type of project. The Git-flow project involves no. of branches.

In this diagram the long running branch labels have solid boarder and the short running branch label have dash labels. You can see the master and the develop branches are the only long running branches. If you look at the master branch you will see there are three commits. Commit A is the initial commit in a repository. We will assume that any commits on the master branch after commit A represent a version of a project that customer can use. These releases have been tagged with version labels. This commit graph includes release a version 1 of the project then surely follows with the minor updates. By keeping things simple we will release version 1 of the project with the single feature named Feature1.

V1.00

V2.00

Master

Hotfix 1

Release 1

**. . . .**

Develop

Feature X

Feature 2

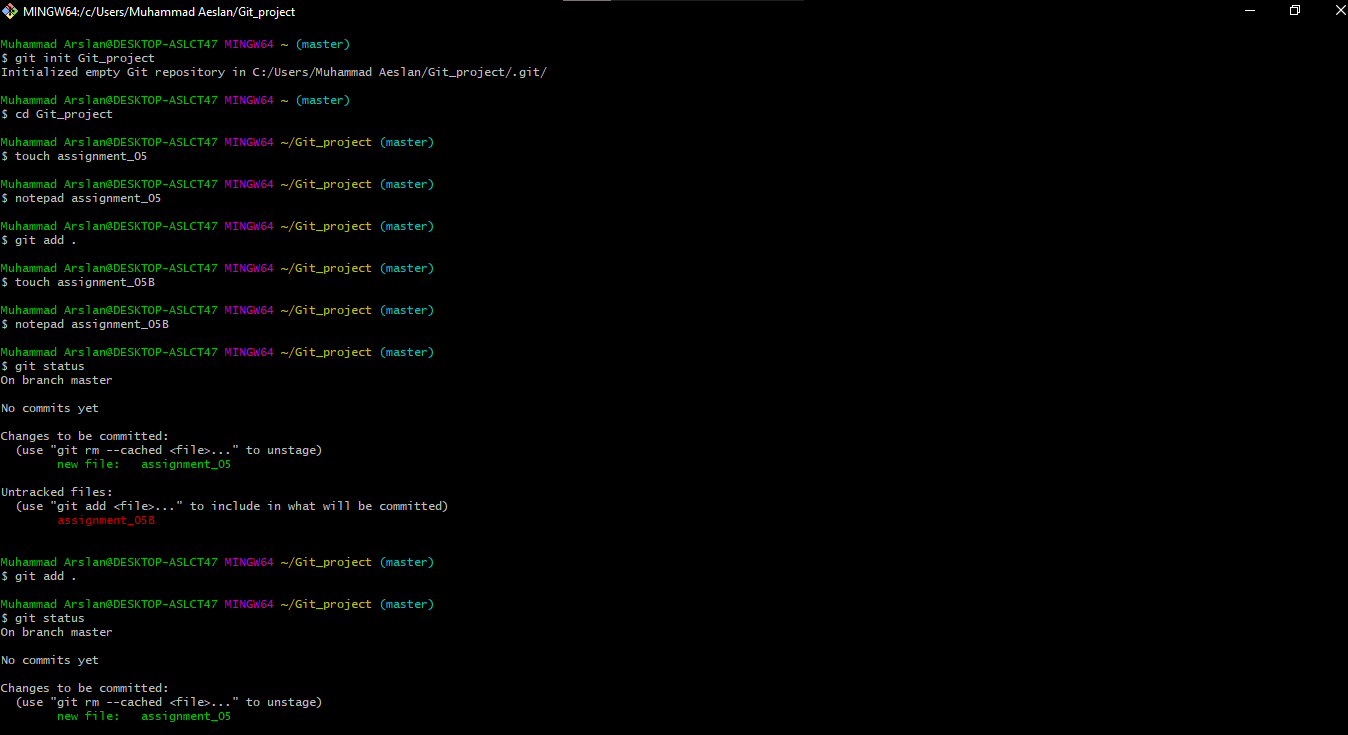
Feature 1

# Q no: 03

**Demonstrate the GIT workflows with any GIT clients?**

**Local Repository:**

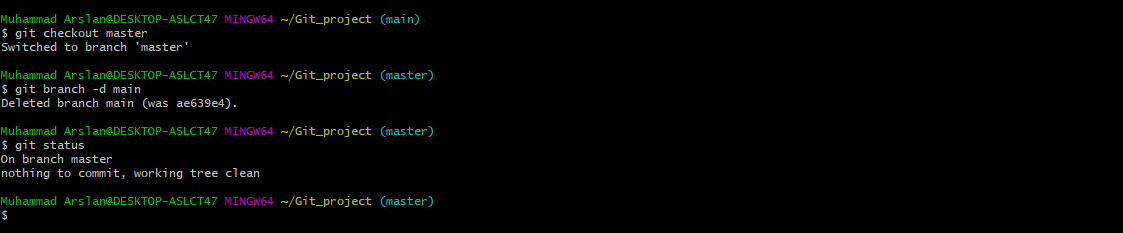
Firstly i have created a local repository by naming it “Git\_project” then i have created two files assignmemt\_05 and assignment\_05B respectively. After making changing in both i have added first file and forget to add the second one. Then i have checked the status for where i am standing then i come to know that i have some uncommitted file to add. Then i add the file using "git add ." comment and again check the status. Now my files are successfully added into local repository

****

After that i have created a new branch for getting hands on feature branching workflow. At present we are at master branch and checkout for the main branch. Making changing in the file i save it and again checkout for the master branch. Now it’s time to merge main branch with master branch.

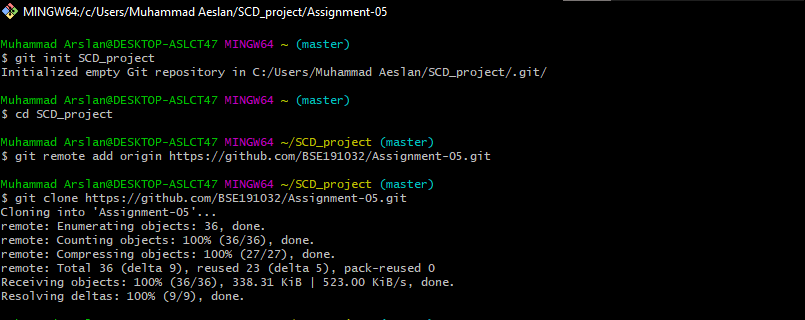
****

After merging both the branches i have deleted the feature branch "main" and check the status that showed we are at master branch and nothing is left to commit.

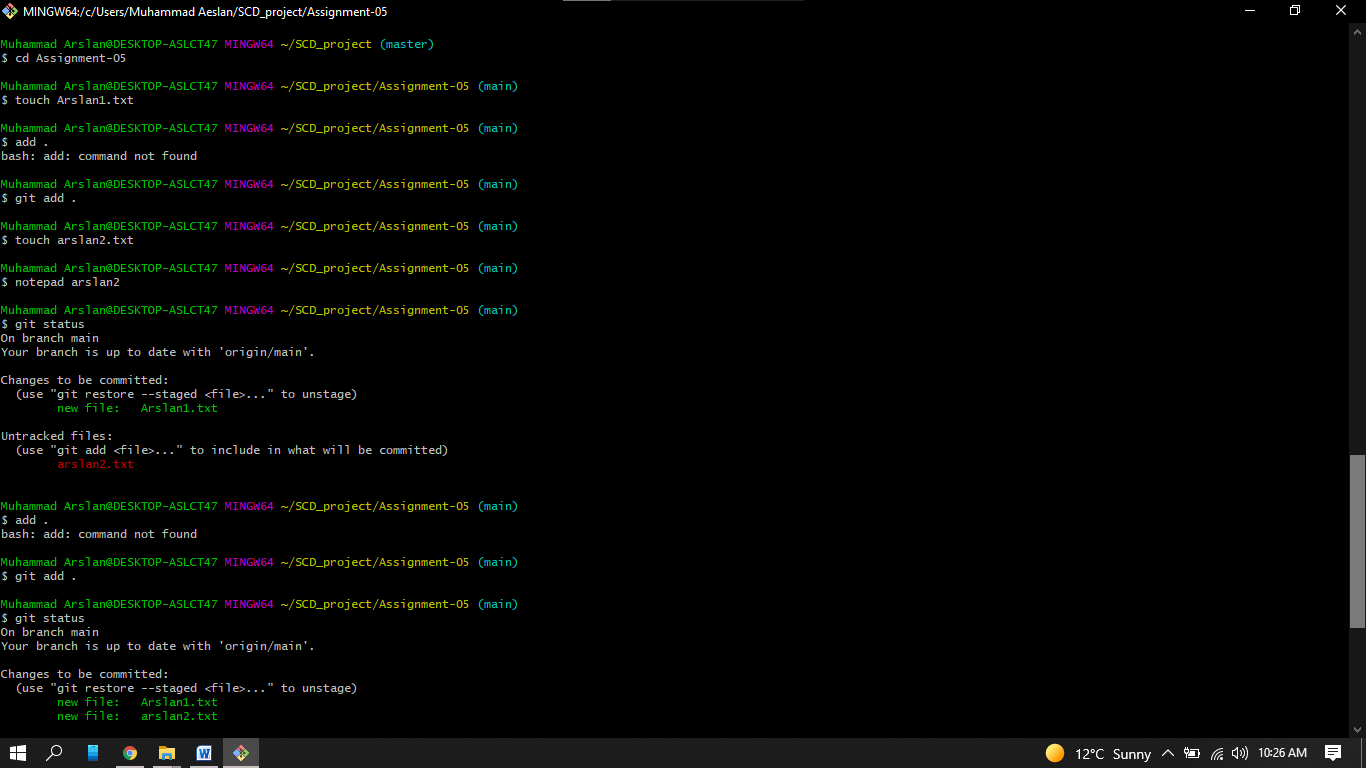
****

* **Centralized workflow**
* **User: 01**

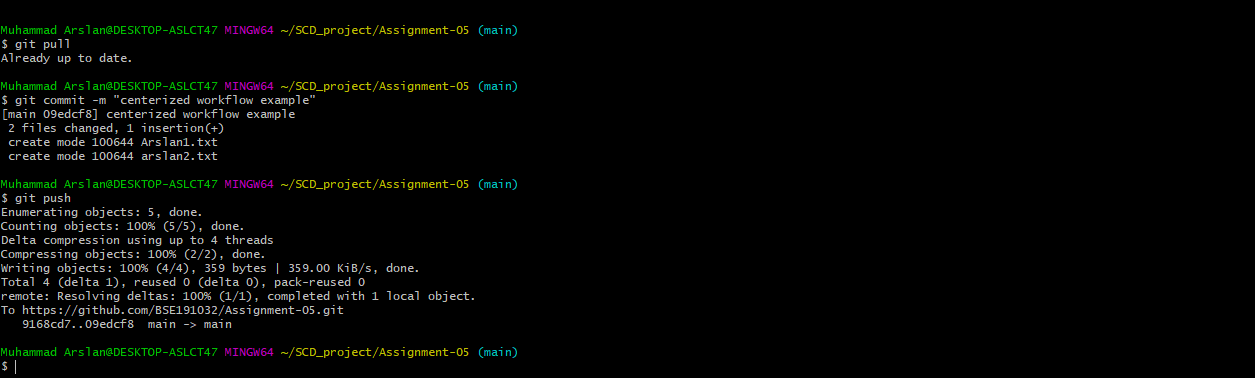
User: 01 “Muhammad Arslan” has created a remote repository and cloned it using URL <https://github.com/BSE191032/Assignment-05.git>

****

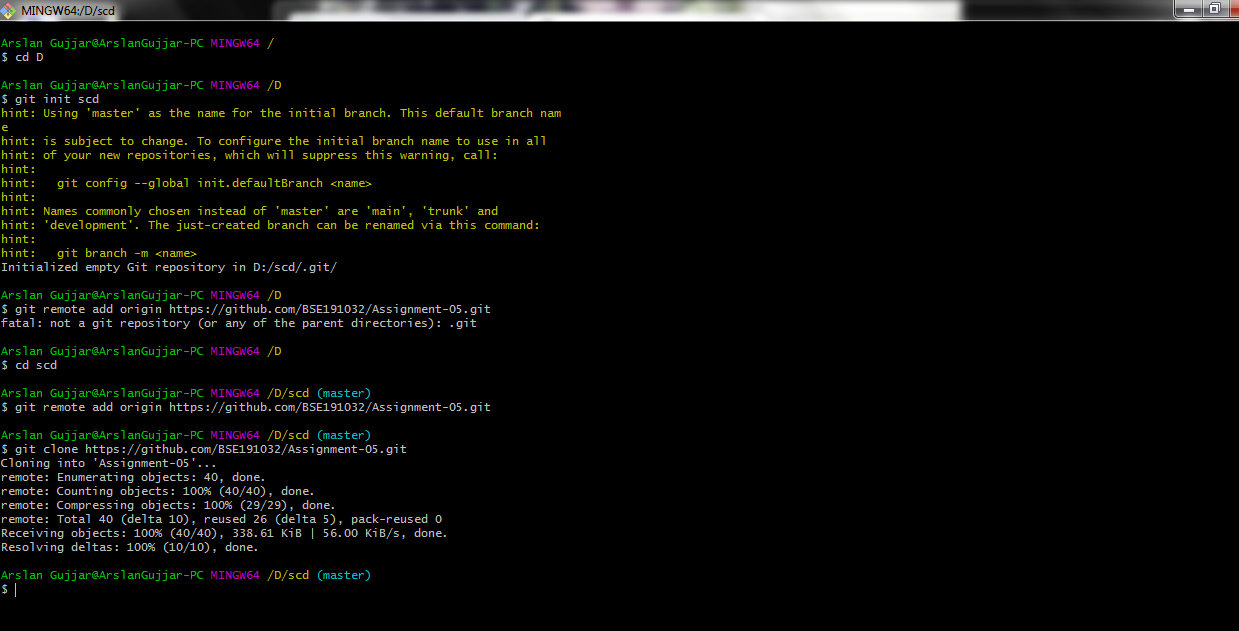
After cloning from remote repository, first I went into Assignment\_05 and created two files Arslan1.txt and Arslan.txt respectively. After making changing in both i have added first file and forget to add the second one. Then i have checked the status for where i am standing then i come to know that i have some uncommitted file to add. Then i add the file using "git add ." comment and again check the status. Now my files are successfully added.

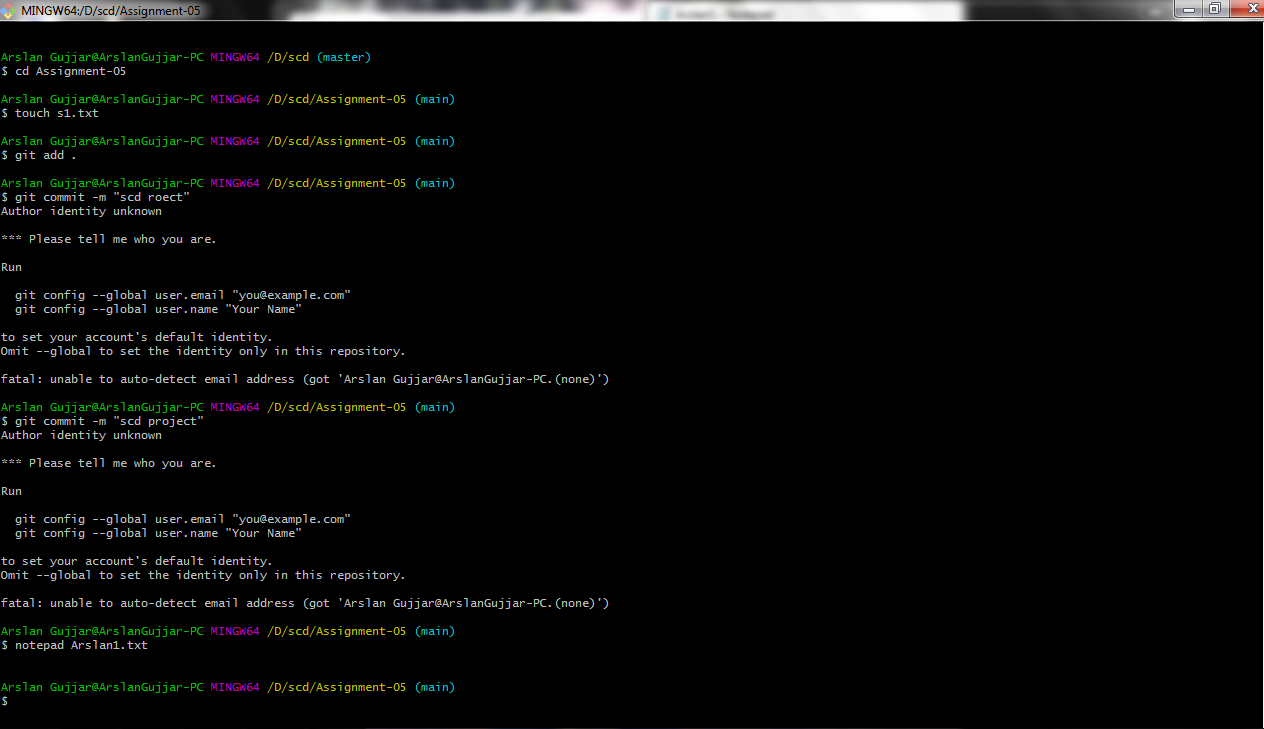
****

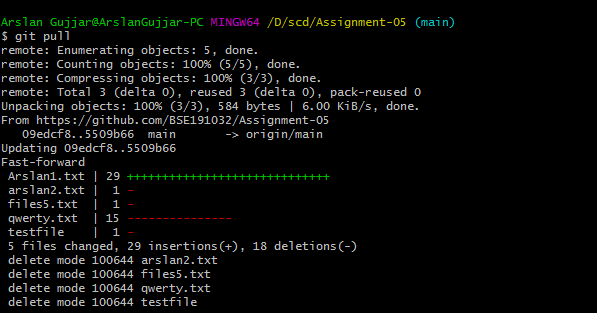
Before committing my update to the remote repository I just generate pull command that showed me a message “Already up to date” means user: 02 haven’t yet committed anything by now so that mean I can commit and push my updates to the remote repository without any conflict.

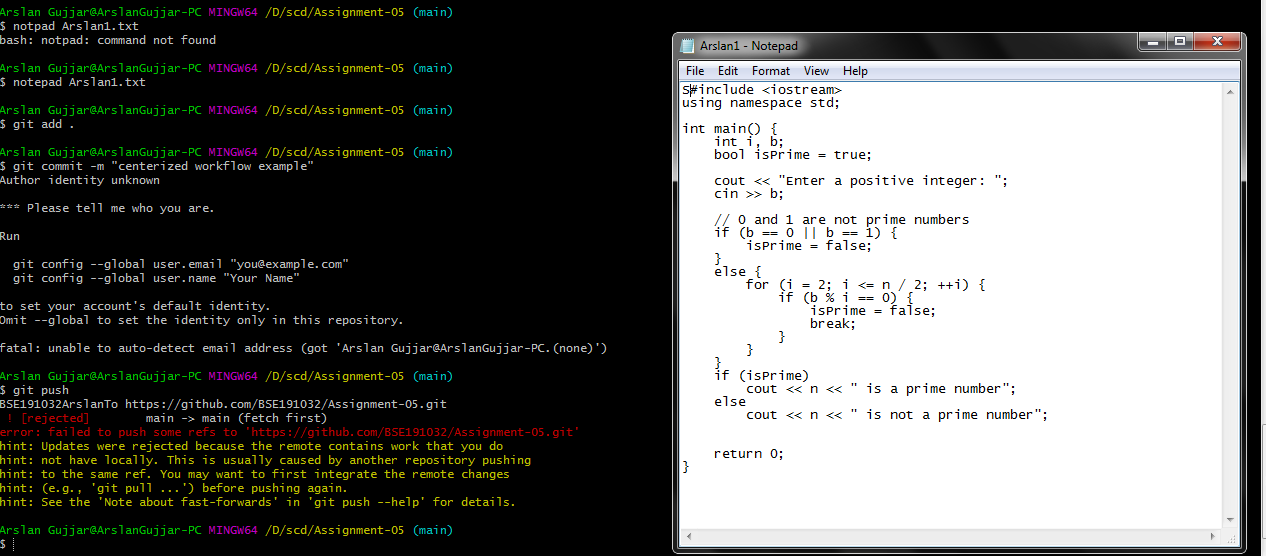
****

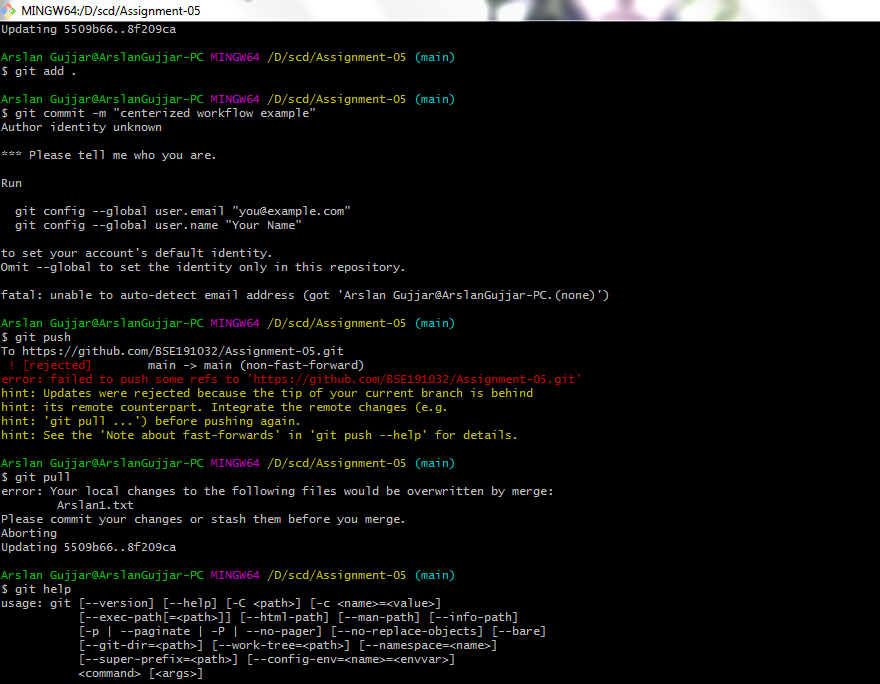
* **User: 02**

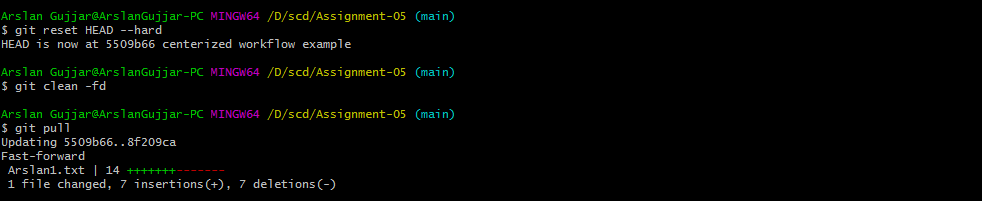
****

****

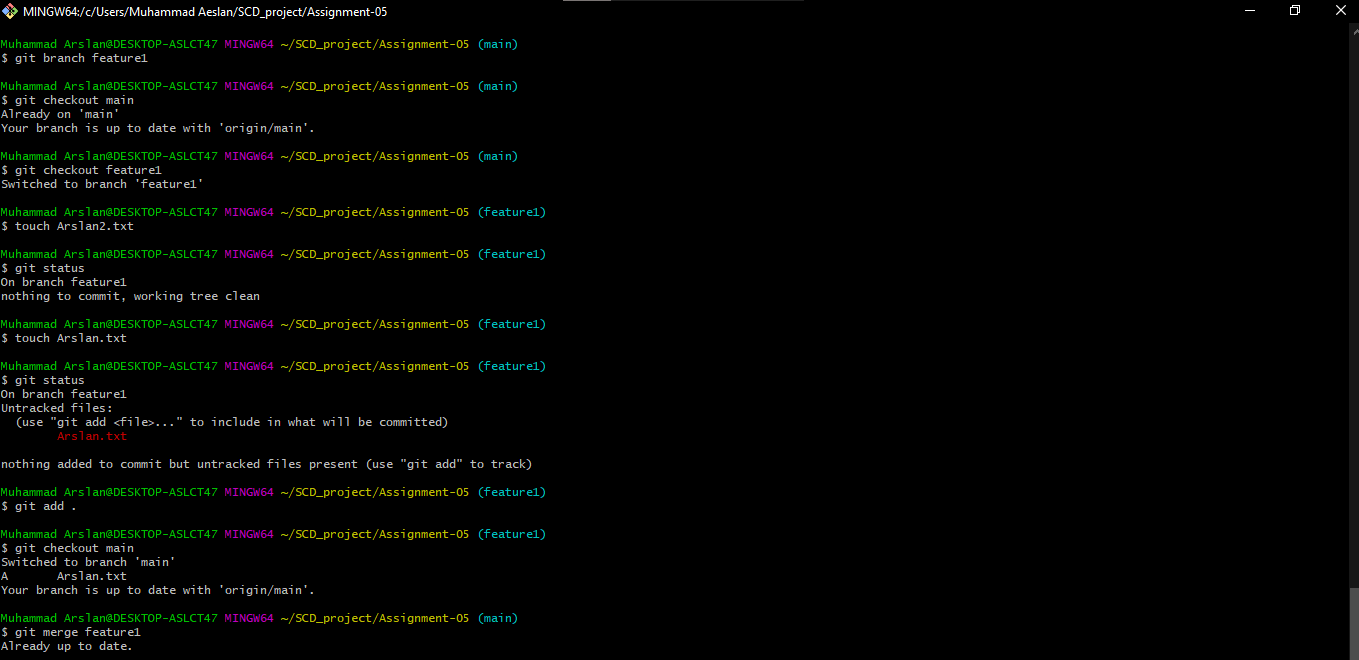
****

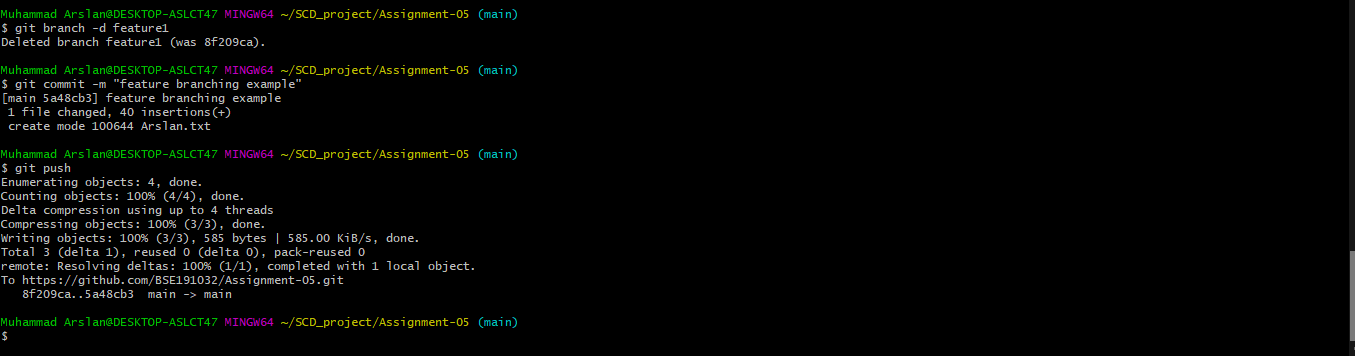
****

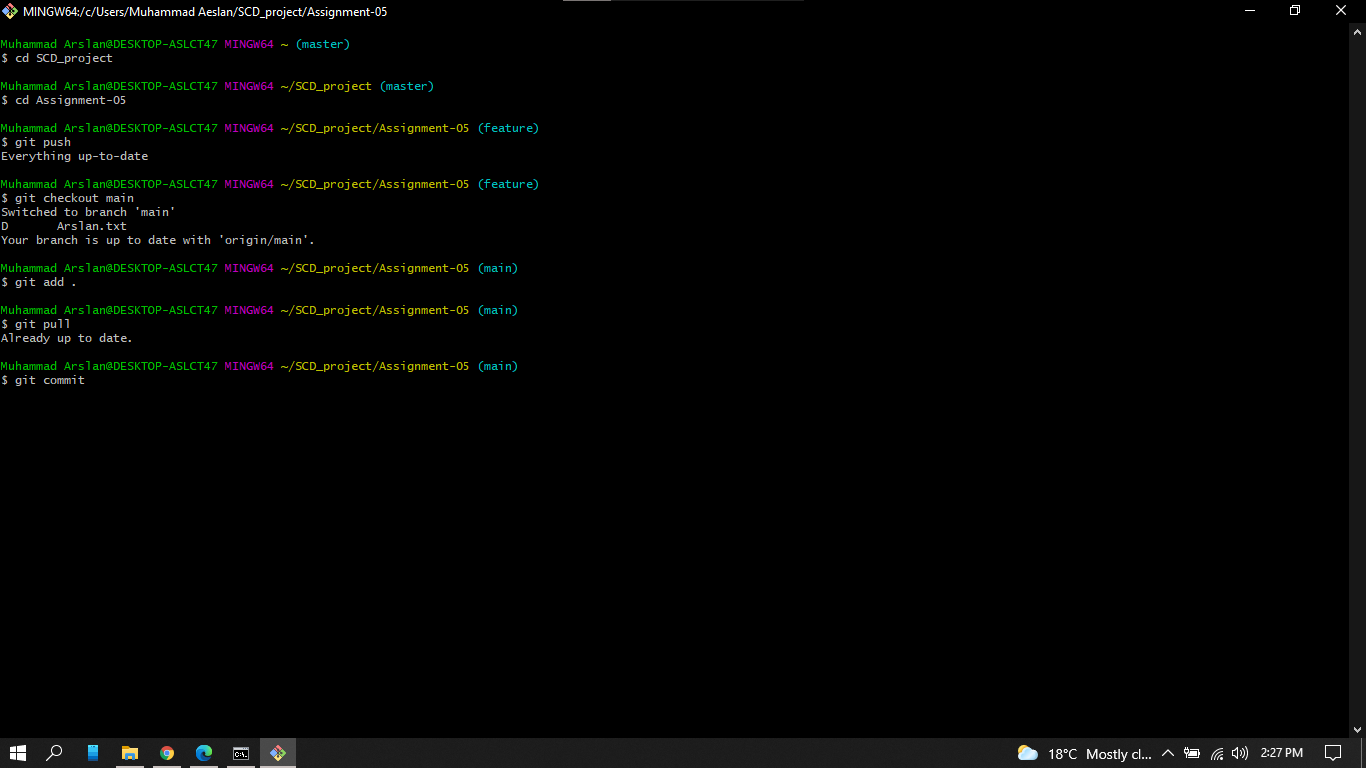
****

****

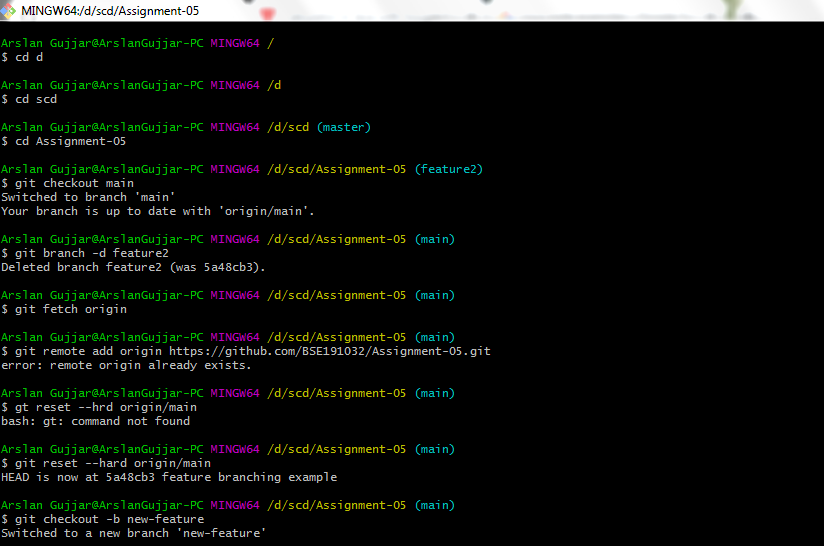
**Feature branching user: 01**

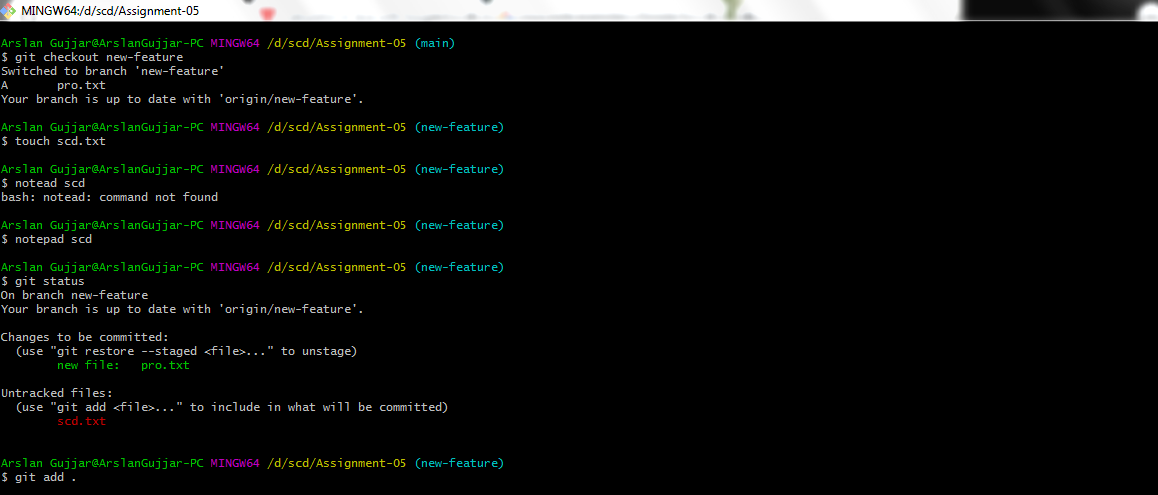
****

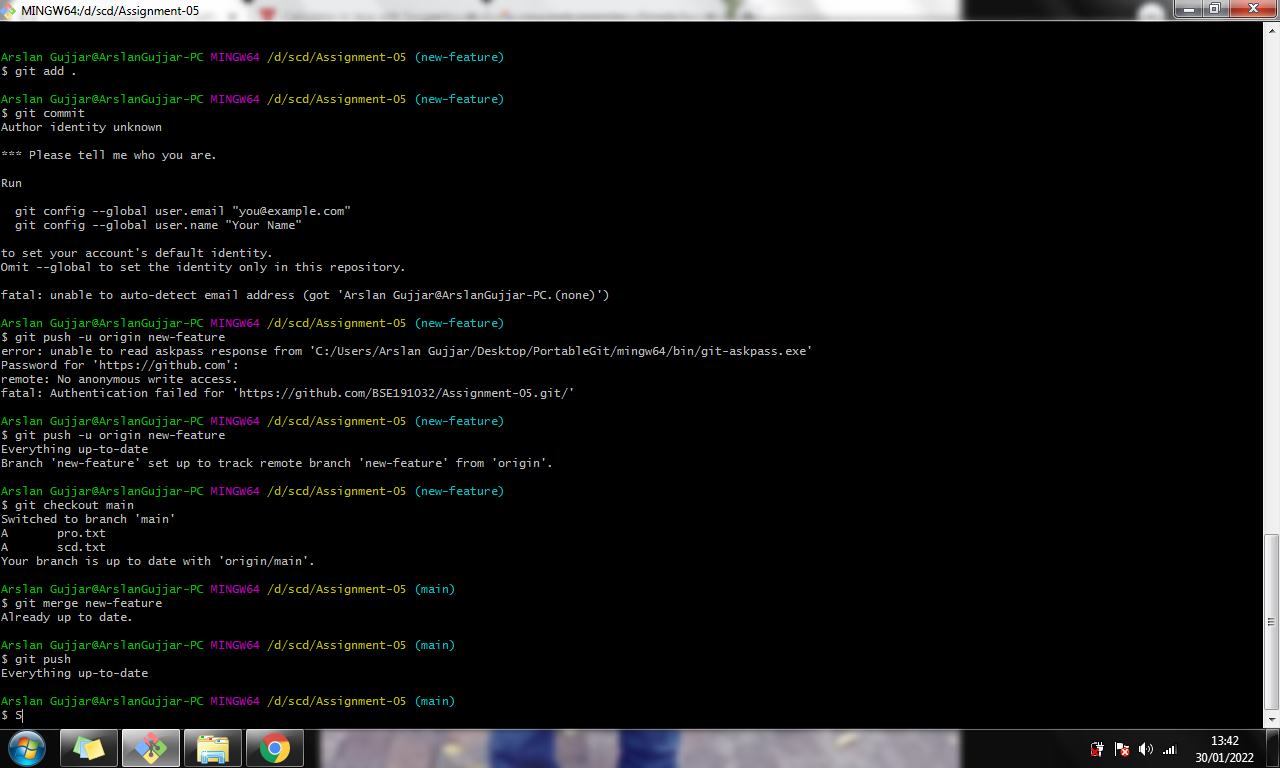
****

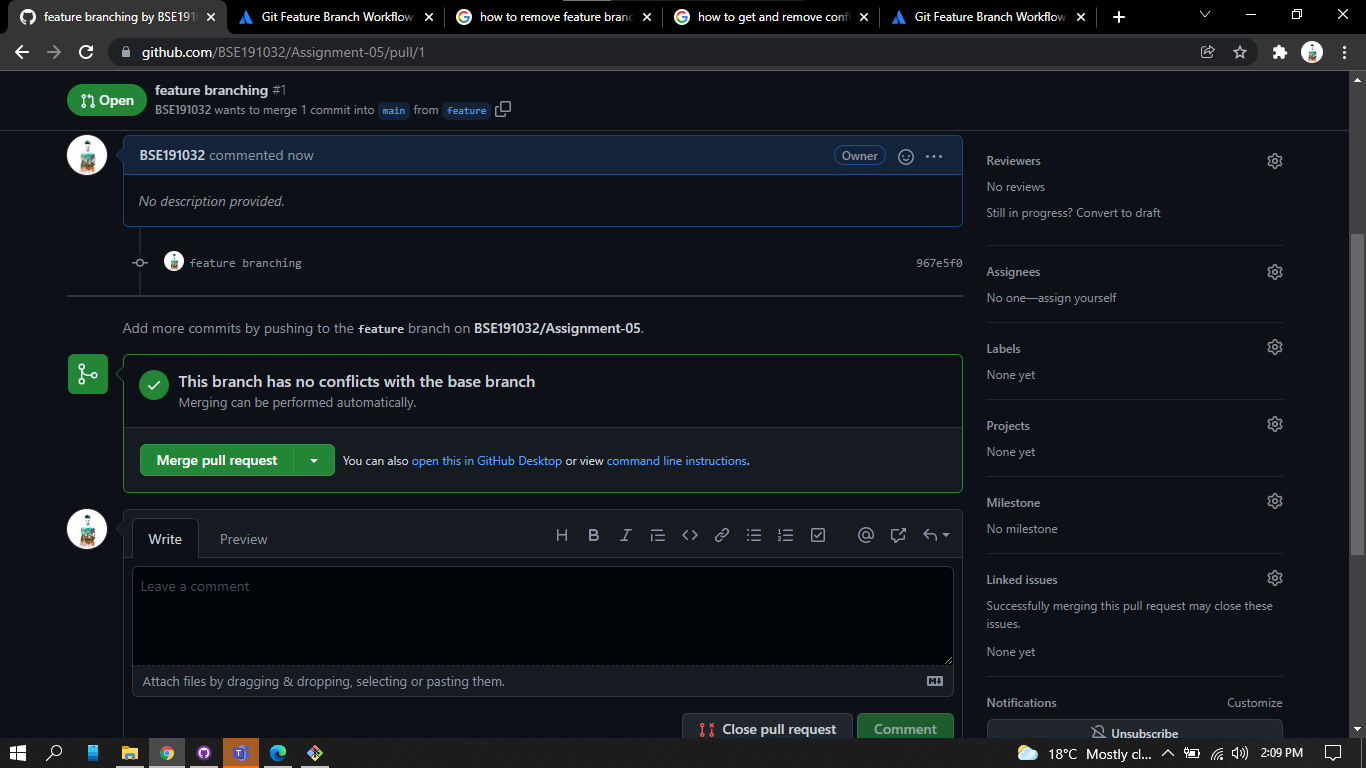
****

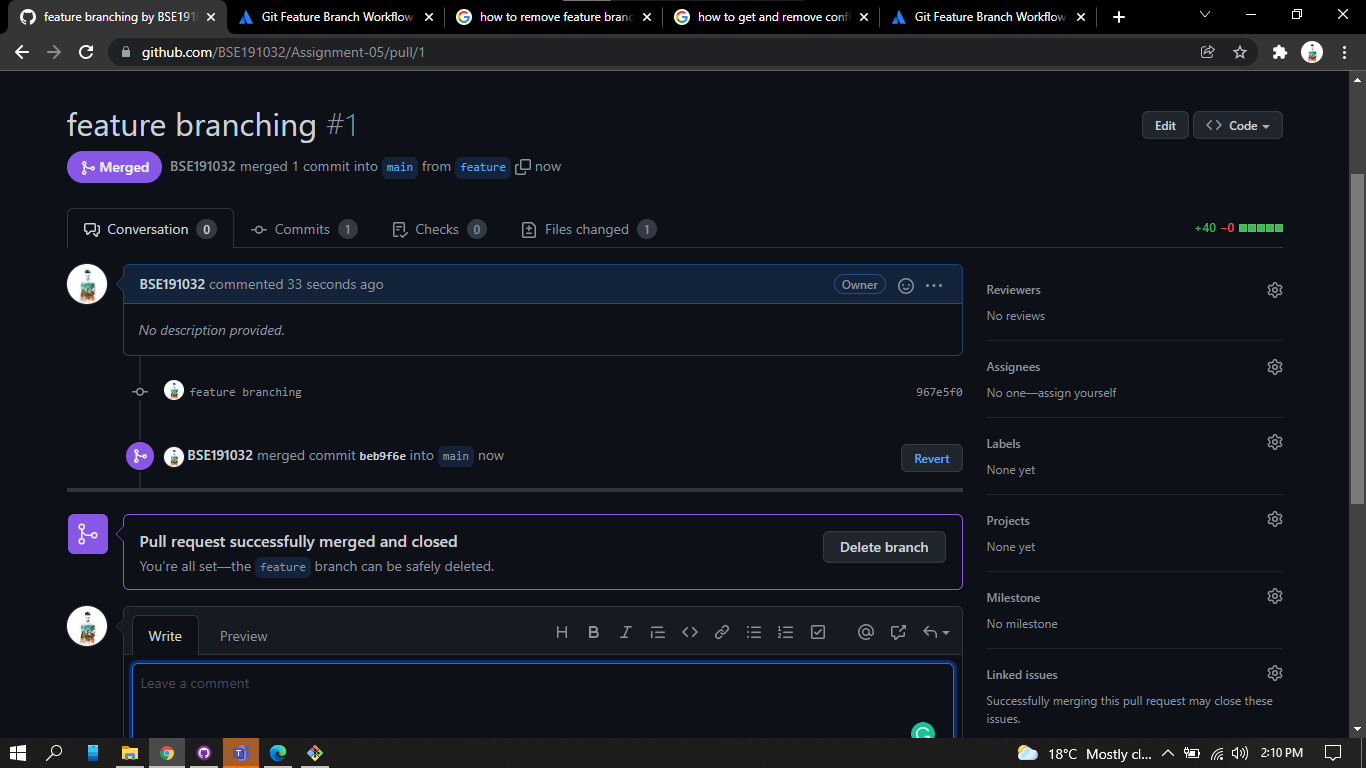
**User: 02**

****

****

****

****

****