**DEVOPS (Merged - CSIZG514/SEZG514)(S2-20)**

**Assignment 1**

*Submitted by-*

**Jharana Sahu (2020MT93743)**

**Nishchita TS (2020MT93601)**

**K G Chandramouli (2020MT93683)**

**Santosh Singh Chauhan (2020MT93753)**

**Nirmal Kumar S (2020MT93549)**

1. **Create Repo**

Graphical user interface, text, application, Teams

Description automatically generated

1. **Manager role**

Created roles to manage Reviewer and approver.

Graphical user interface, application

Description automatically generated

1. **Collaborator**

Request sent for collaboration and accepted.

Graphical user interface, application

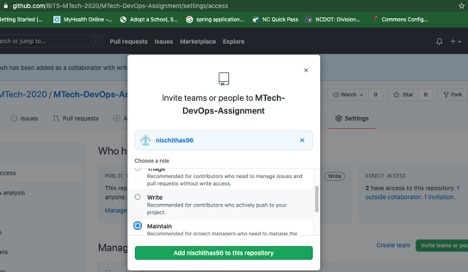
Description automatically generated

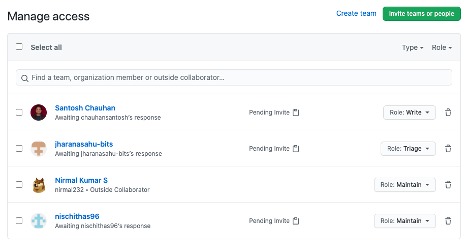
1. **Add team members as collaborators and assign them appropriate rights**

Team members are added with appropriate rights.

Graphical user interface, application

Description automatically generated





1. **Create a branch(development/production/feature)**

Text

Description automatically generated

Graphical user interface, application, email

Description automatically generated

1. **Edit files or create new files followed by commit**

File is created and committed.

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. **Clone the repo and Create pull-request**

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text

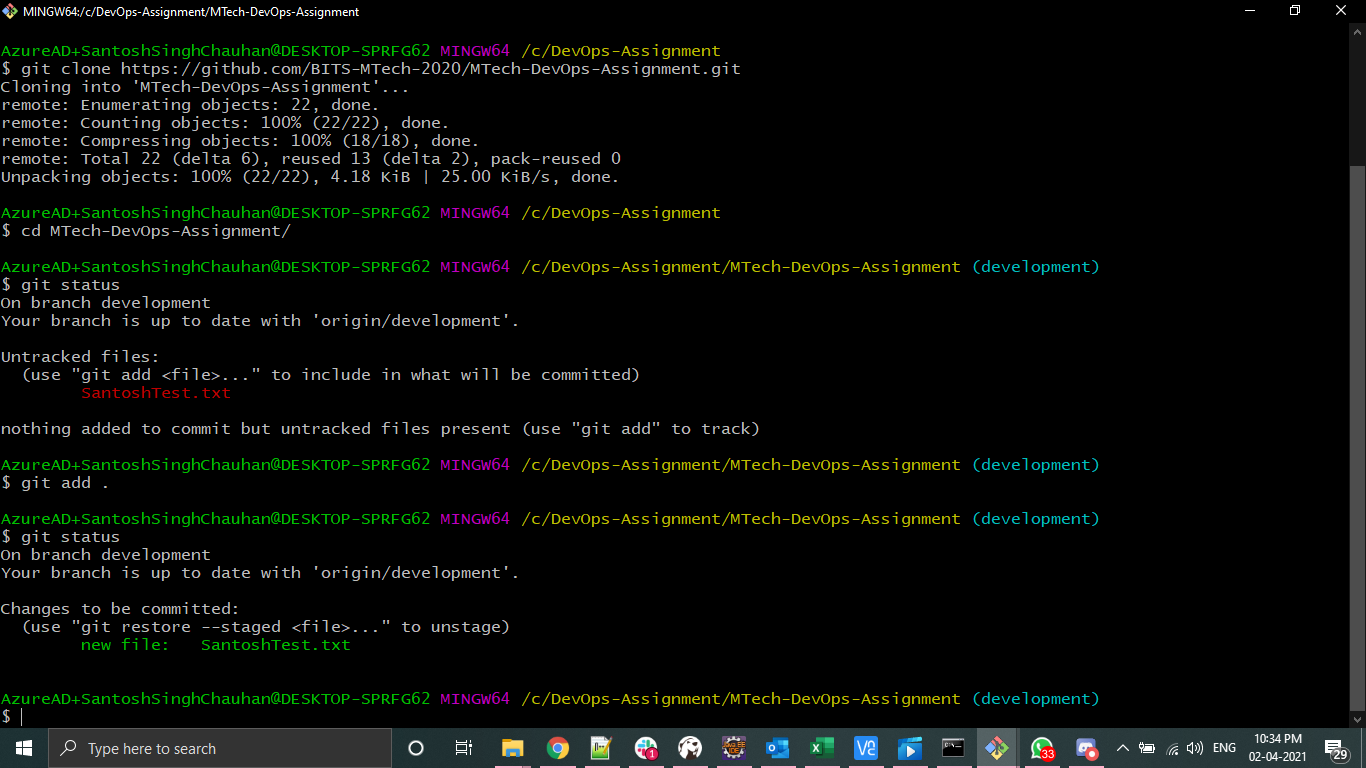
Description automatically generated

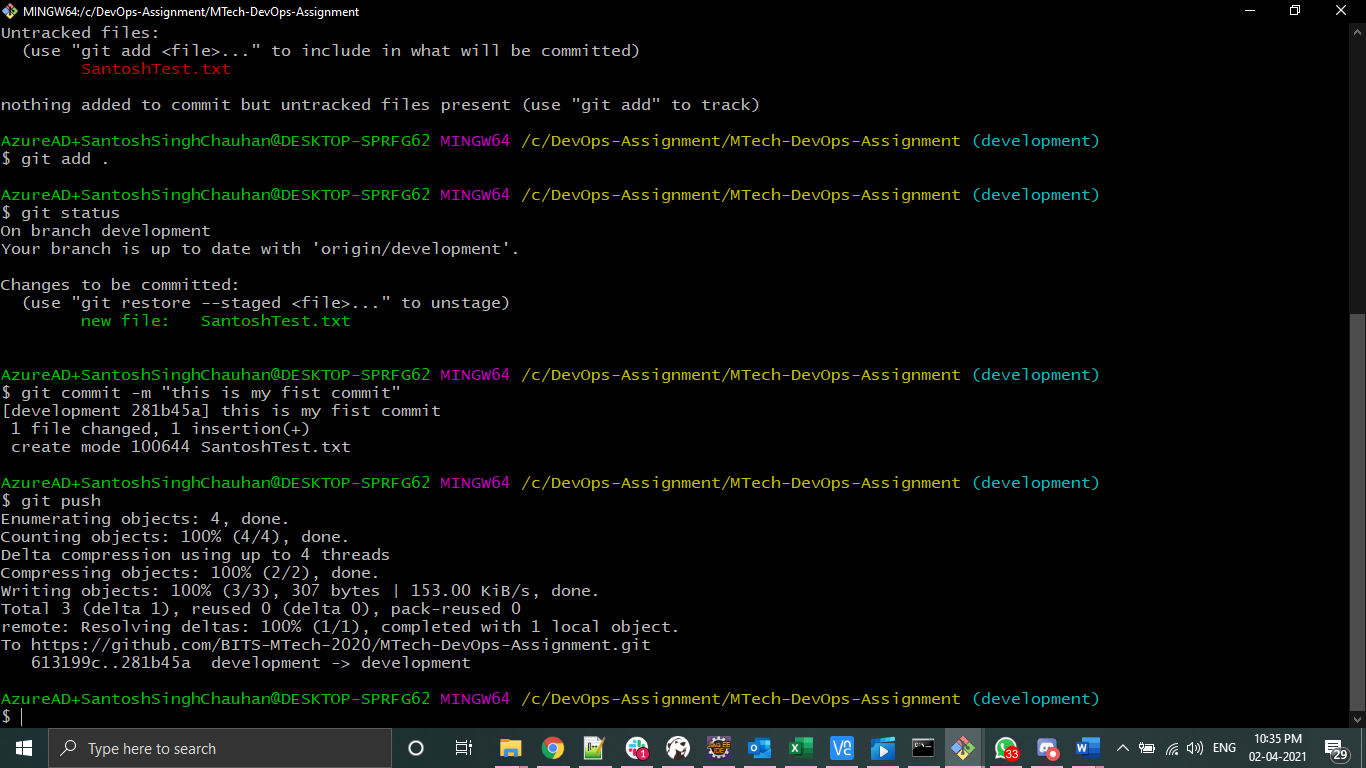
Graphical user interface, text, application, email, Teams

Description automatically generated

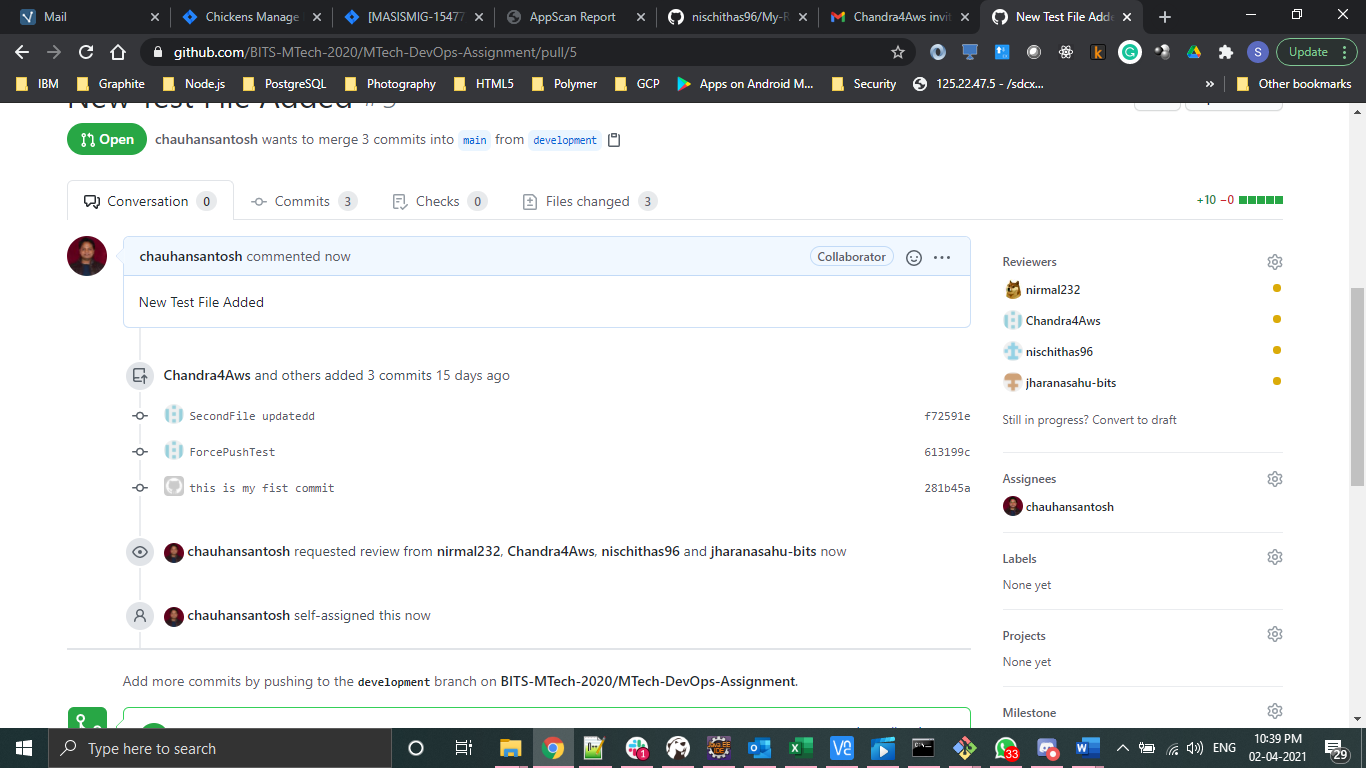
**Pull request demo with team members added as Reviewers**

1. Clone the repo, add a new file, commit and push changes to remote repo.





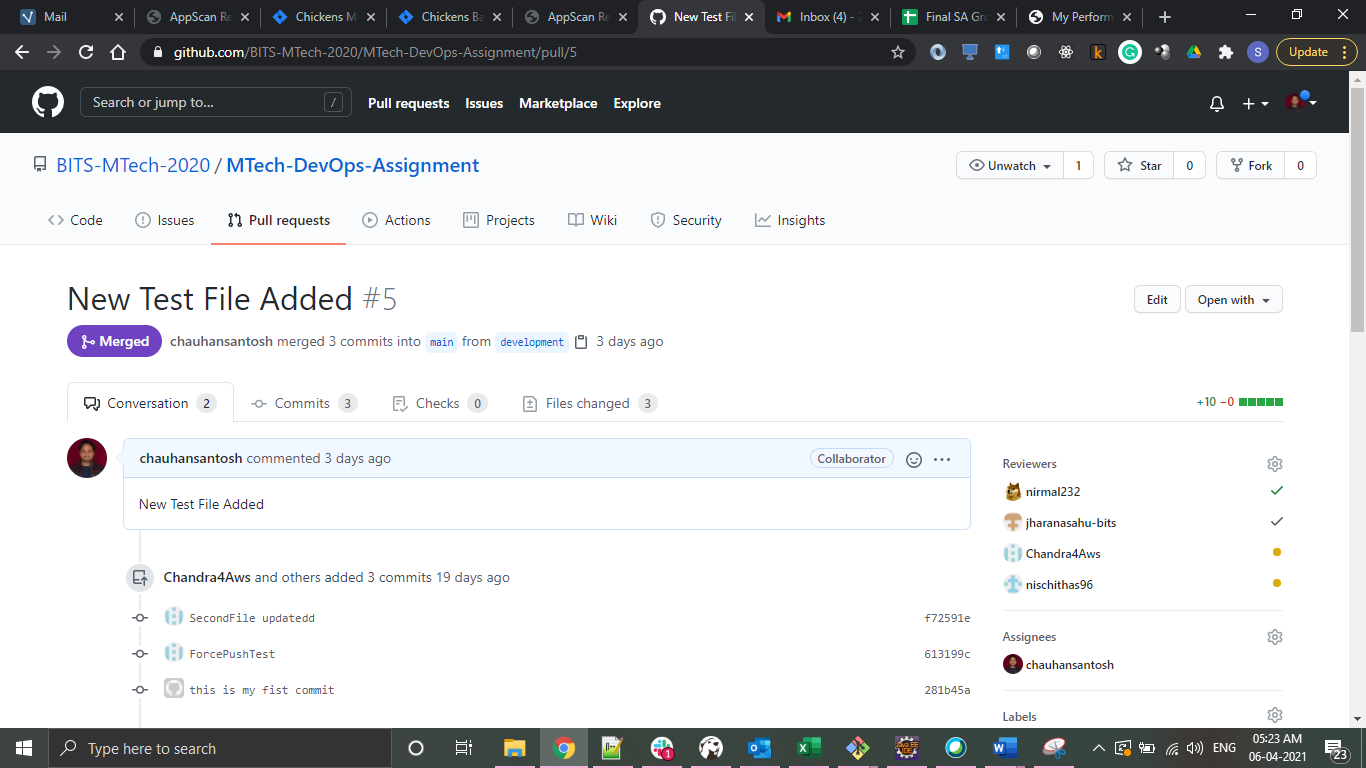
1. Create a pull request and add team members as Reviewers



1. Review requested



1. PR (pull request) approved and merged to main branch from development branch.





1. **While collaborating your work, showcase how conflicts are resolved**
   1. Same file is checkout by two different users.
   2. Modified by both users
   3. First user pushed changes
   4. Second user getting the conflict message on the same while when try to push it
   5. Resolving the conflict manually and pushing the changes.

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

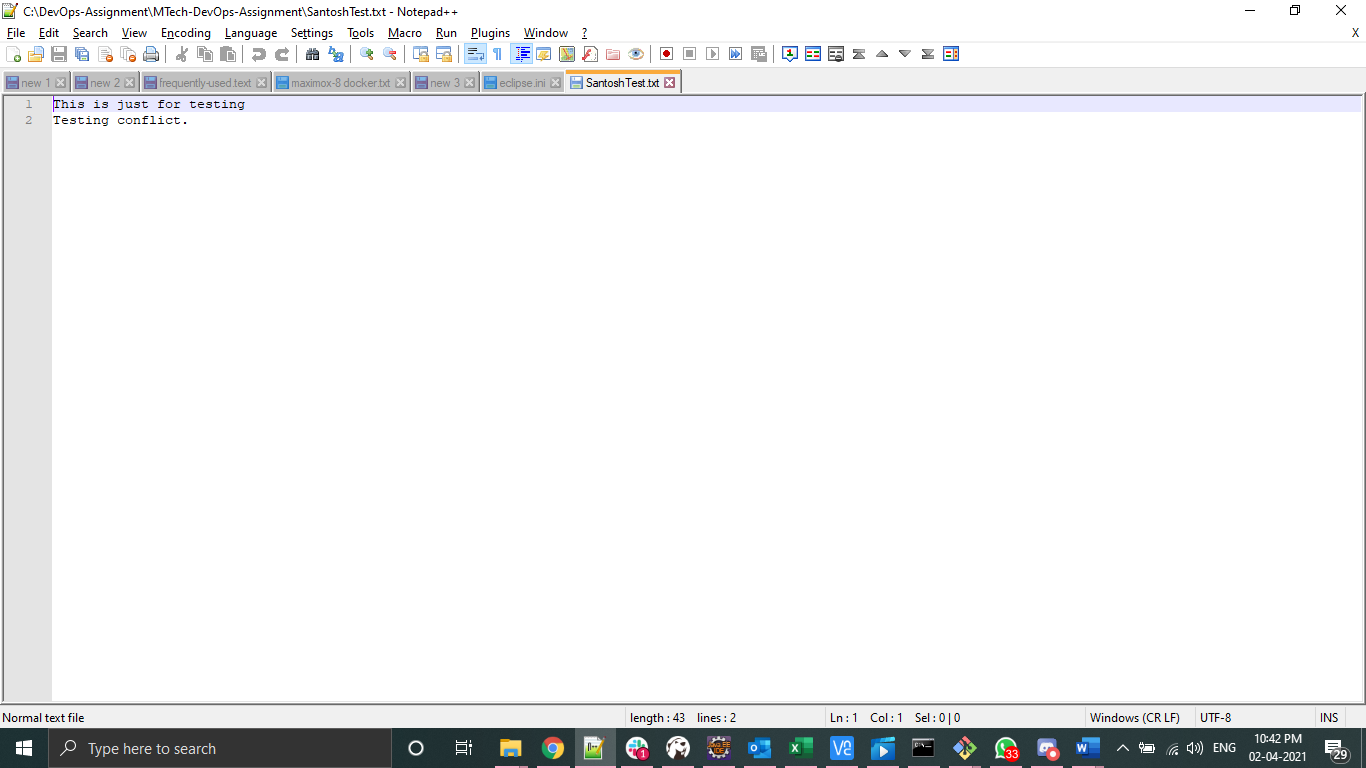
Description automatically generated

Text

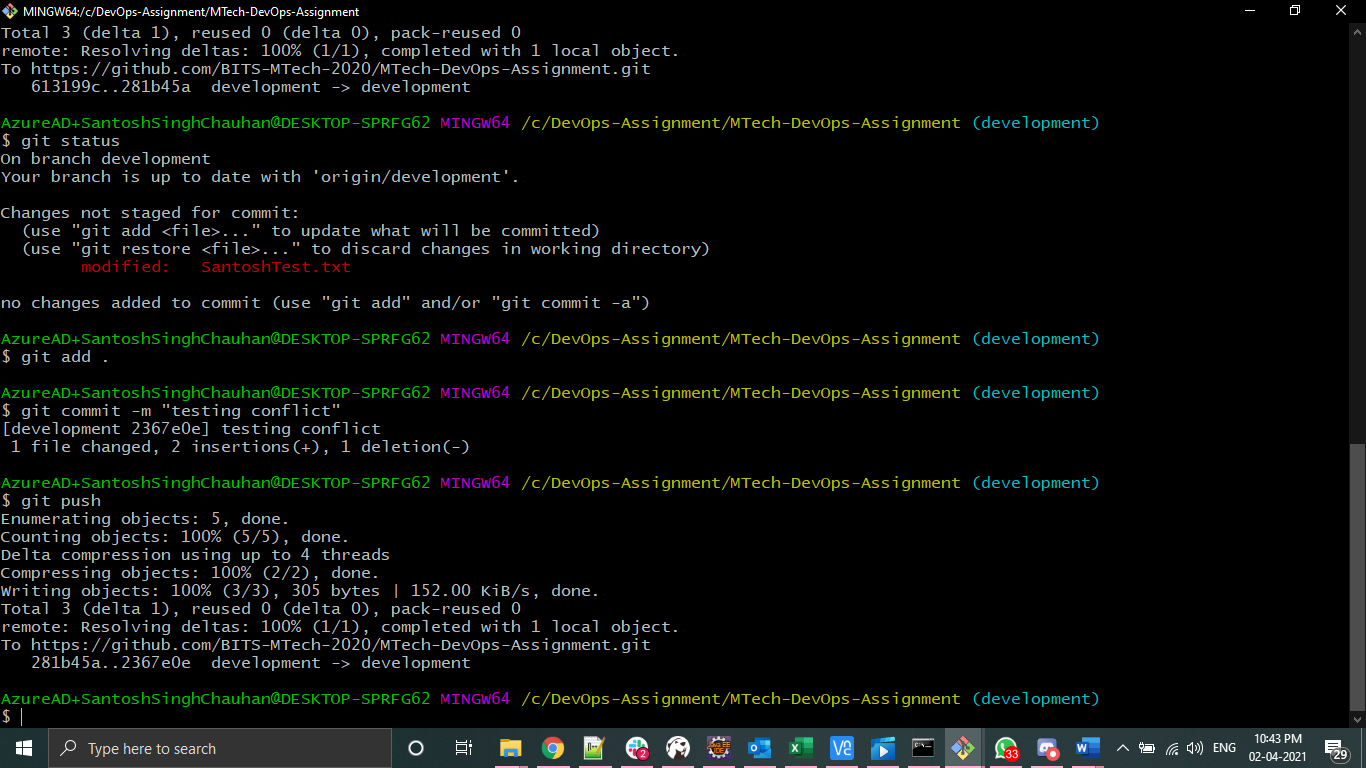
Description automatically generated

**Conflict demo by team members**

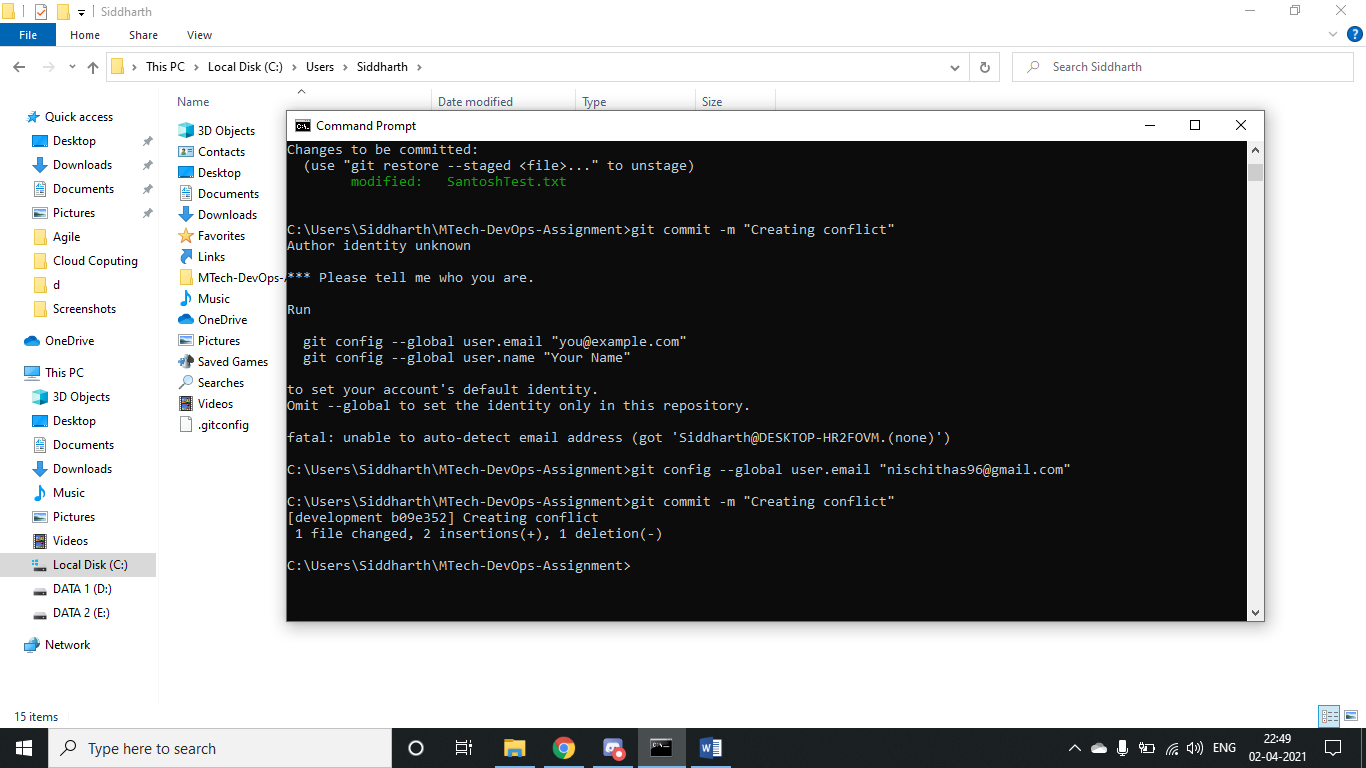
1. Text file modified by a Santosh



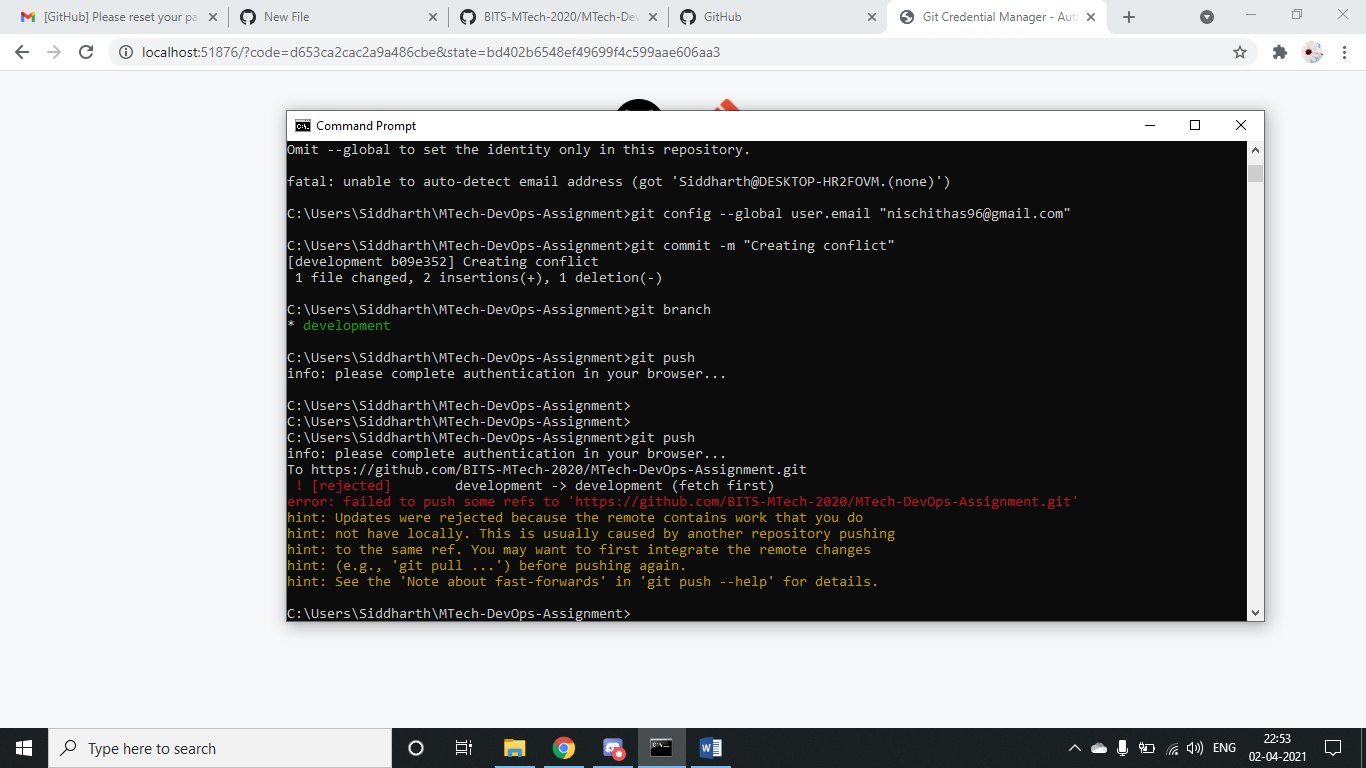
1. Changes committed and pushed to remote repo by Santosh



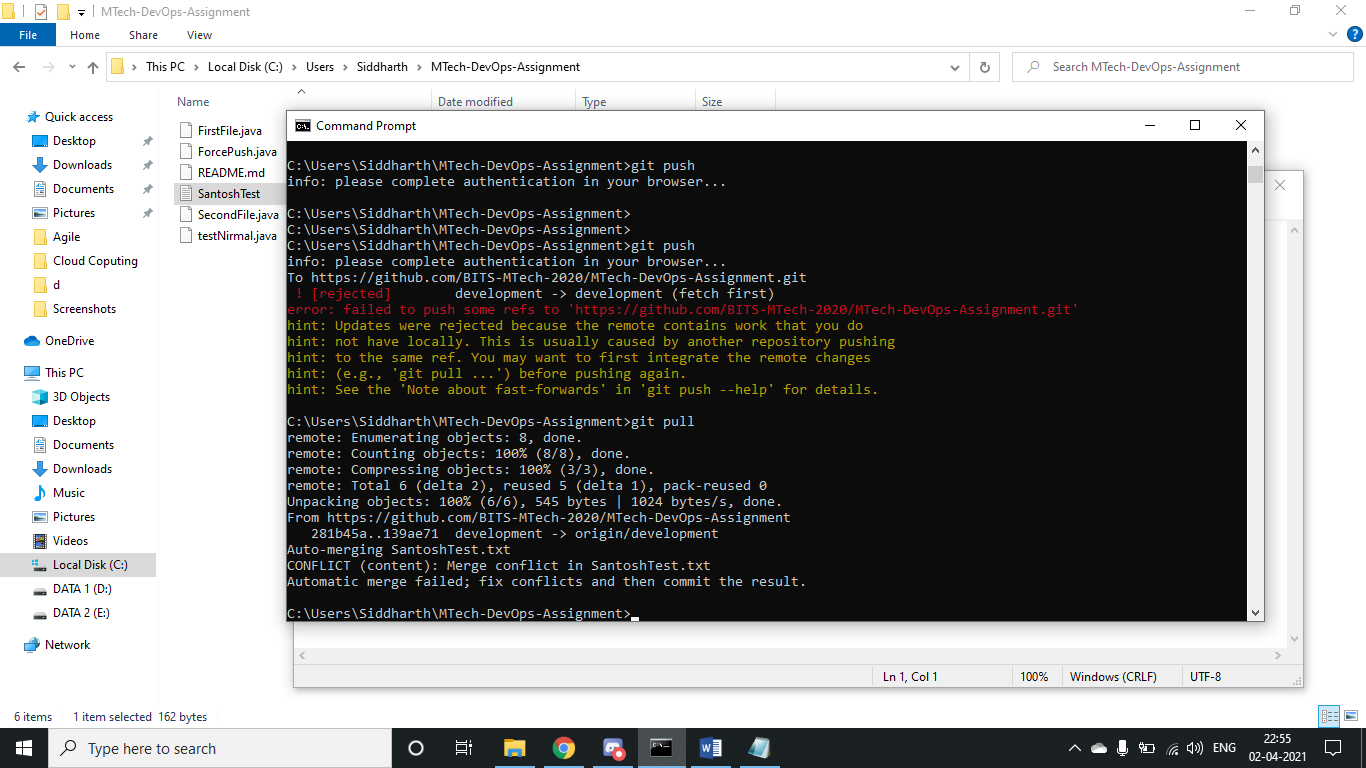
1. Same file modified and committed by Nishchita



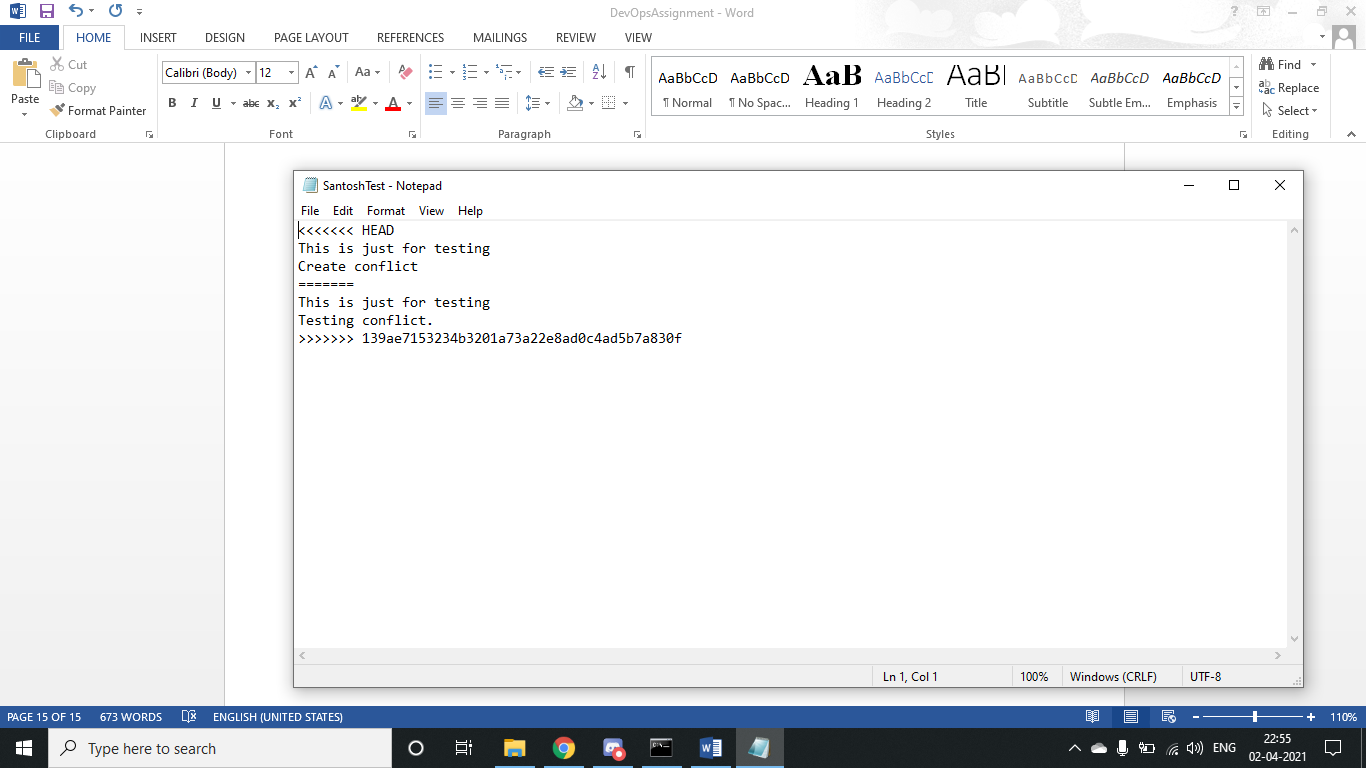
1. Getting conflict error while pushing the changes to remote repo by Nishchita



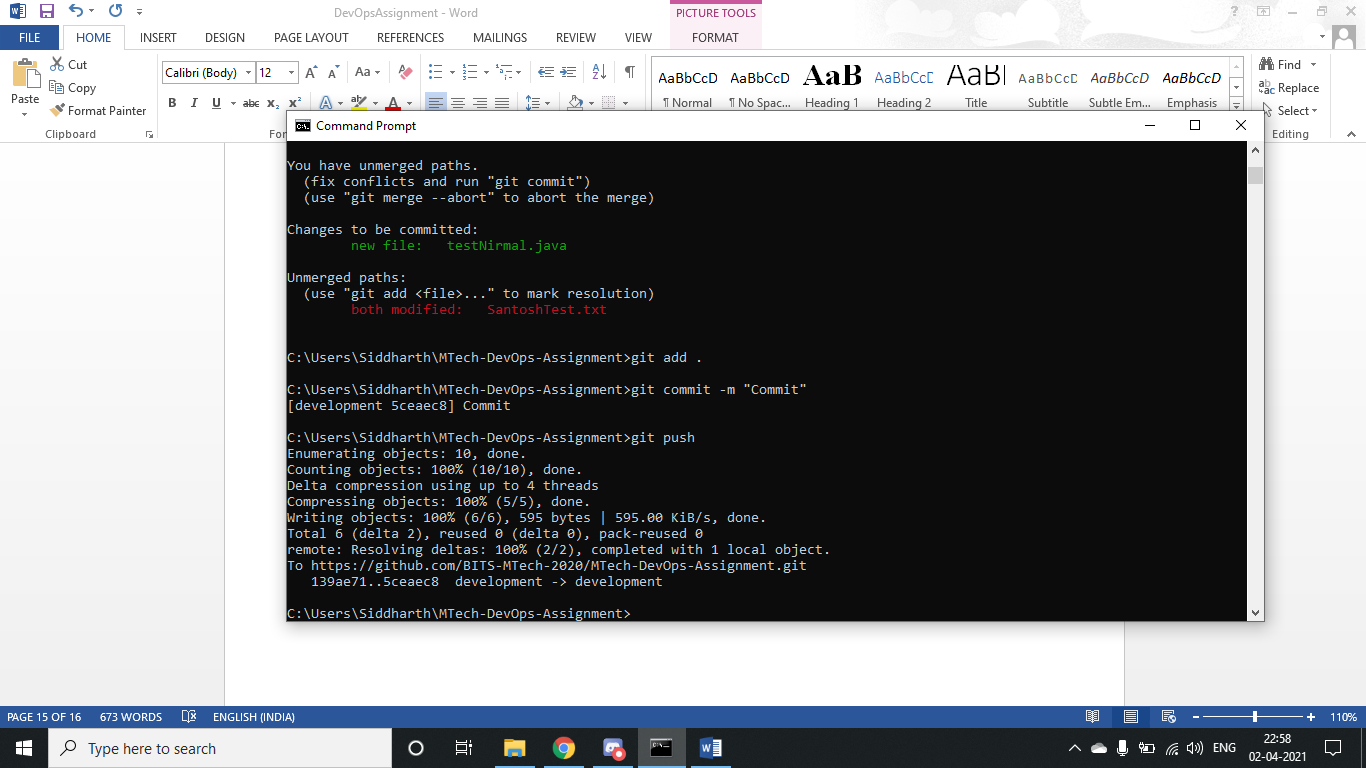
1. Get the latest changes using git pull



1. Opening the file and resolving conflicts manually



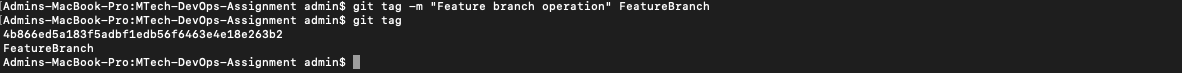
1. Committing and pushing the changes after resolving conflict manually



1. **Create tag such as open issue, or feature-added**

Tag is created with commit id as well with description





Text

Description automatically generated

1. **Do a force push and then later reset the changes**

With conflict force push is done and did the reset.

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. **Stage “development branch to production branch”**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. **Showcase how features are released in versions (merging production to master branch)**
2. Click on “Create a new release” shown below.



1. Publish release after attaching the related binary file

Graphical user interface, text, application, email

Description automatically generated

1. Pre-Release is ready

Graphical user interface, text, application, email

Description automatically generated

1. **state importance of Readme and gitignore files and their usage while working in a distributed environment.**

**README** is provides a general information about software/project that in the same directory/repository of the project. README is used to explain the details of the project at high level. It will have about what is the project used ? What are the dependencies, how to install, how to compile, or how to run it in local. How to debug if there is any issues. What is the current status of the project, is it just an experimental, already in production, or just still on under development.

We put all of the information in the README. So when any engineer/developer reads project, they can understand the project without having much knowledge about the code and no need to look to the source code, what is the project used for, like the function or the endpoint (if REST API projects), etc.

Importance of README

1. To identify and track the developer/engineer progress
2. Gives realistic visibility and development stages to consider to contribute or usage of the project.
3. It will help to get contribution from others to make meaningful project/product.

**Gitignore** is intentionally untracked files to ignore by git.

When we make commits in a git repository, we can choose/decide which files to stage and commit by using git add FILENAME and then git commit. This file will help us to avoid committing unwanted files accidentally commit them (especially if you use **git add .** to stage all files in the current directory). At this point of time **“.gitignore**” file comes in handy to have control committing the files. Git know that it should be ignored listed files and not to track them for any changes.

In a distributed environment, there can be many more configuration files specific to their local development environment and other static config files specific to their development environment which is not required to push it to Central repository. Using the .gitignore, developer/user can list down all their files which need not to pushed and use only for local or specific purpose only. In this way, only the required files can be pushed to central repository, a clear segregation of the files.

Additionally, there is no need of any steps to make run the application on individual development environment to tweak the config files before and after GIT commit. Also there is no need of change of config or other files during the central build and deployment process.

1. **Pull Requests are linked with e-mail to the manager who finally approves the changes.**
   1. Notification email to the approver for review.

Graphical user interface, text, application, email

Description automatically generated

* 1. Git screenshot showing raised for review.

Graphical user interface, text, application, Teams

Description automatically generated

* 1. Notification email to the user about the pull request.

Graphical user interface, text, application, email

Description automatically generated