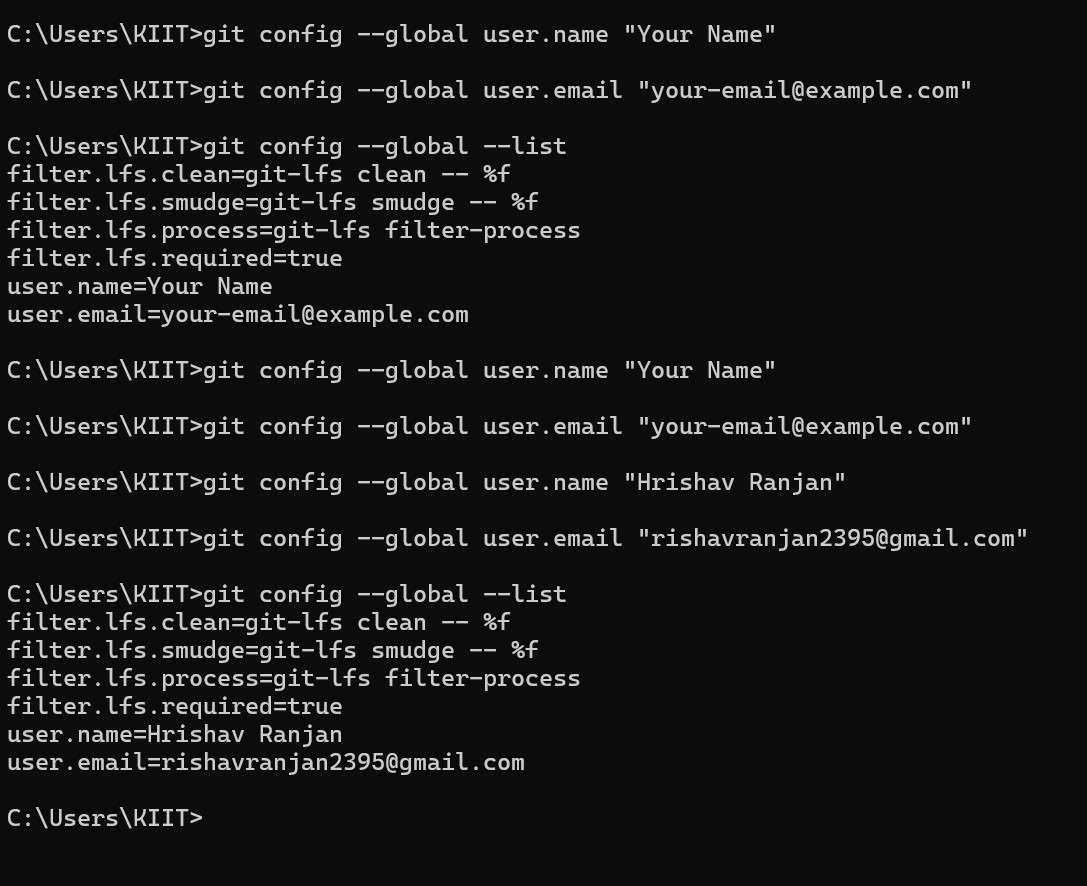
**Skill:-GIT**

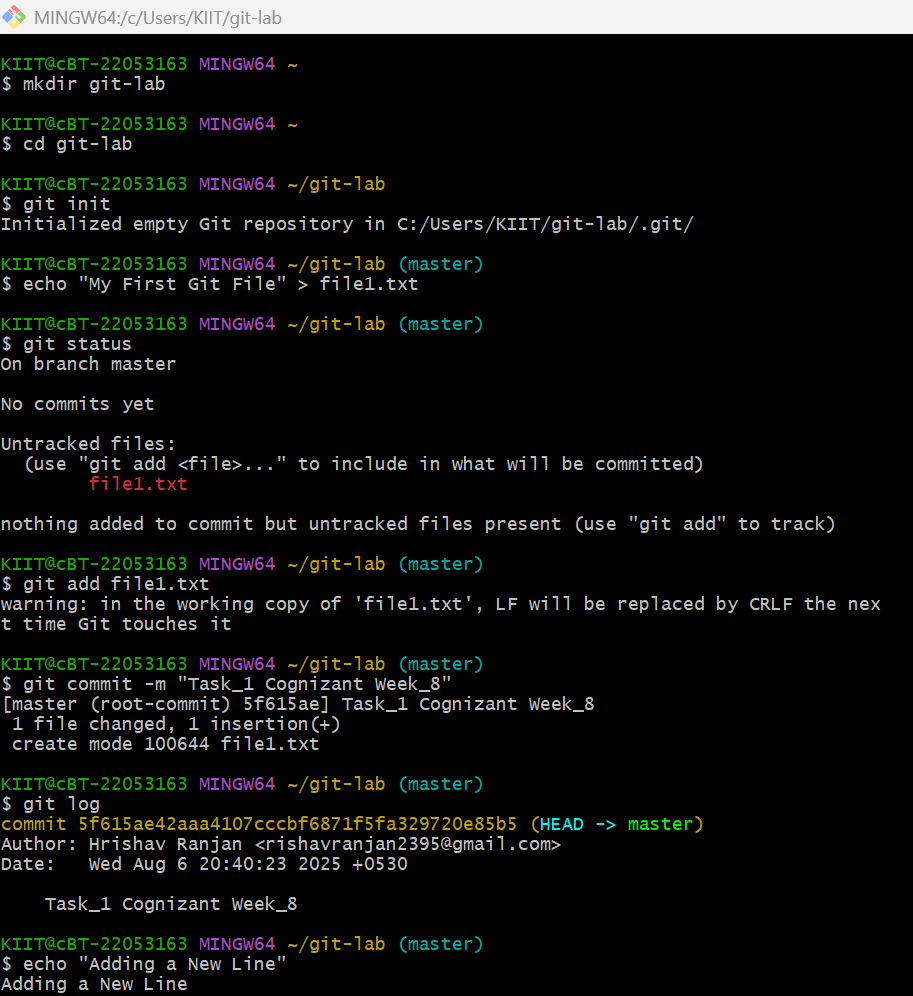
**Filename:- 1. Git-HOL**

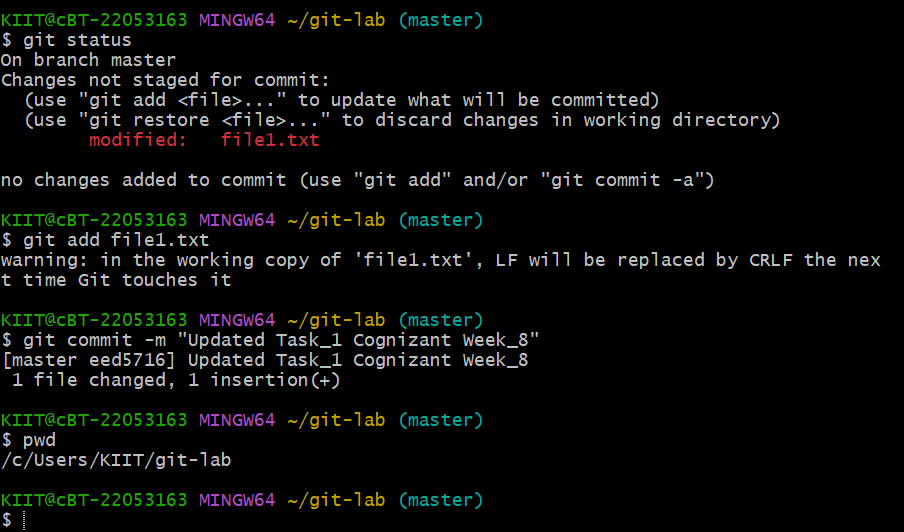
**Solution:-**

***Setup our machine with Git Configuration***

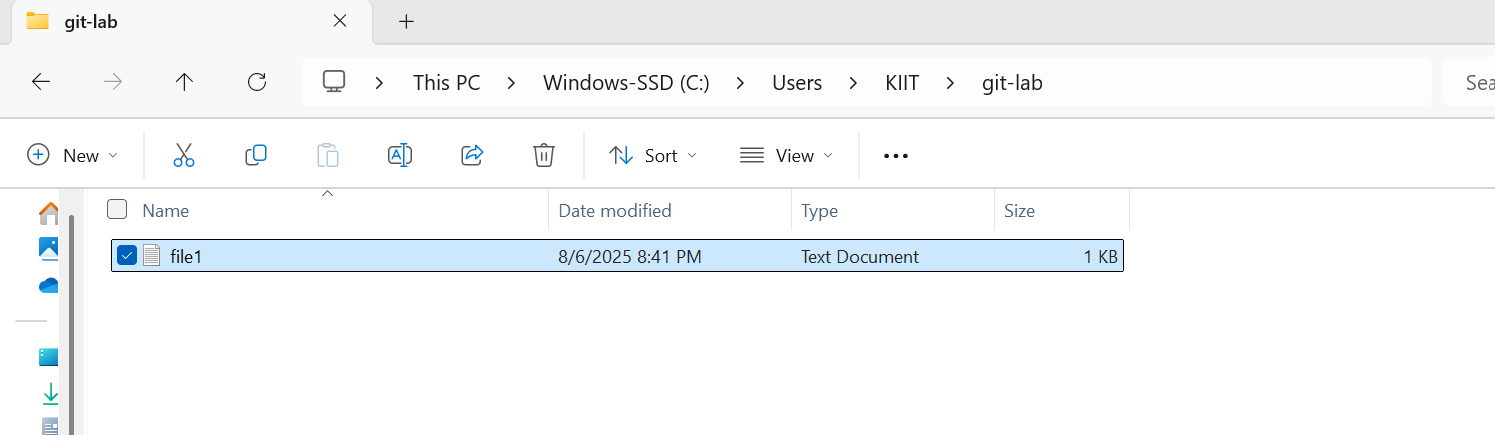


***Adding a file to source code repository***

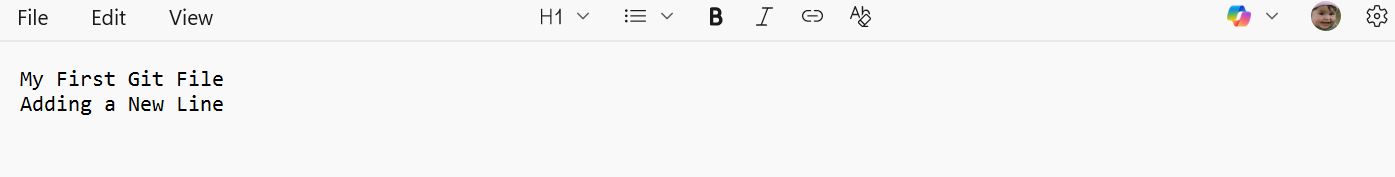




***Screenshots of files on local system***



***The content of Notepad***



**Skill:-GIT**

**Filename:- 2. Git-HOL**

**Solution:-**

**1. Explain git ignore**

**Answer:-**

.gitignore is a **special file** used to tell Git **what files or folders to ignore** — meaning **not to track or commit** those files in the repository.

**2. Explain how to ignore unwanted files using git ignore**

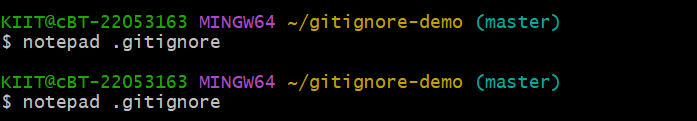
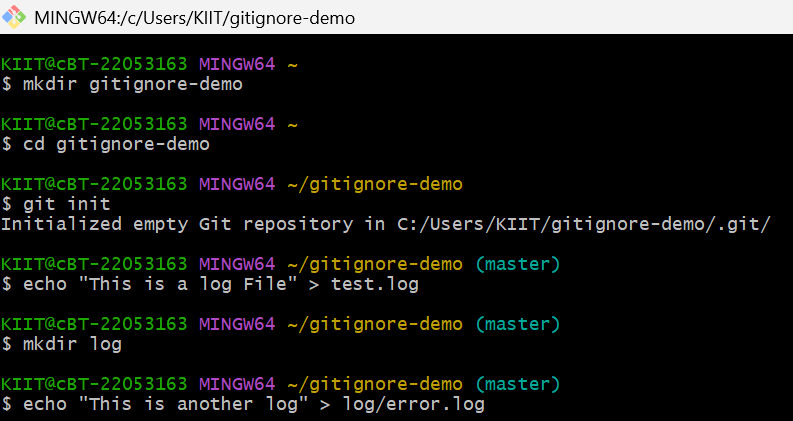
**Answer:-**

To ignore unwanted files or folders in your Git project:

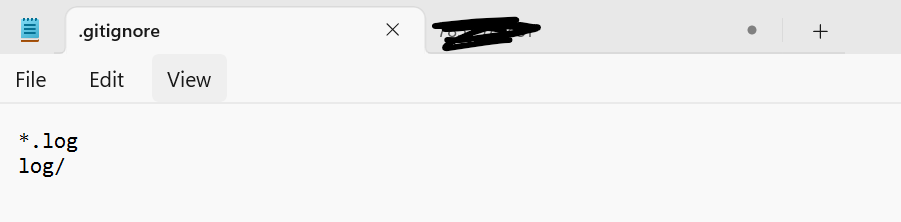
- Create a file named .gitignore in your project directory.

- Add patterns or file names you want to ignore.

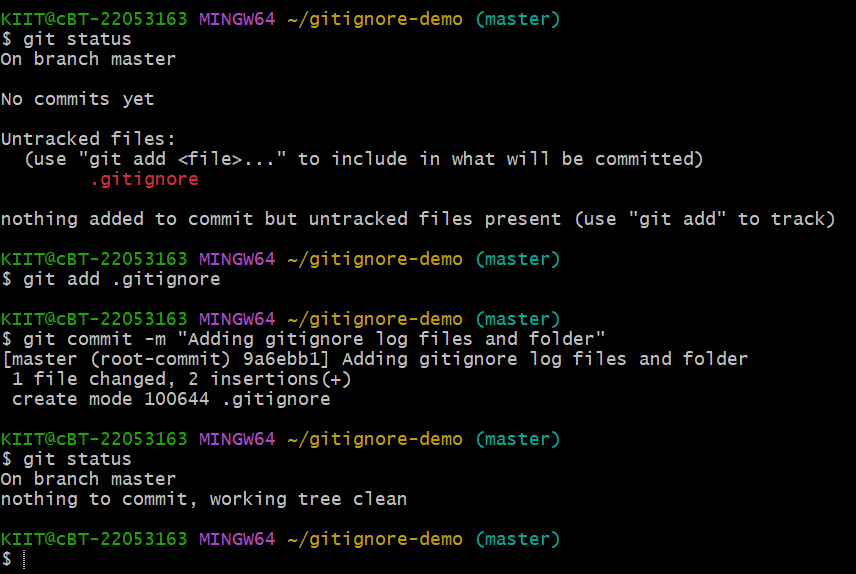
***Created a “.log” file and a log folder in the working directory of Git. Update the .gitignore file in such a way that on committing, these files (.log extensions and log folders) are ignored.***



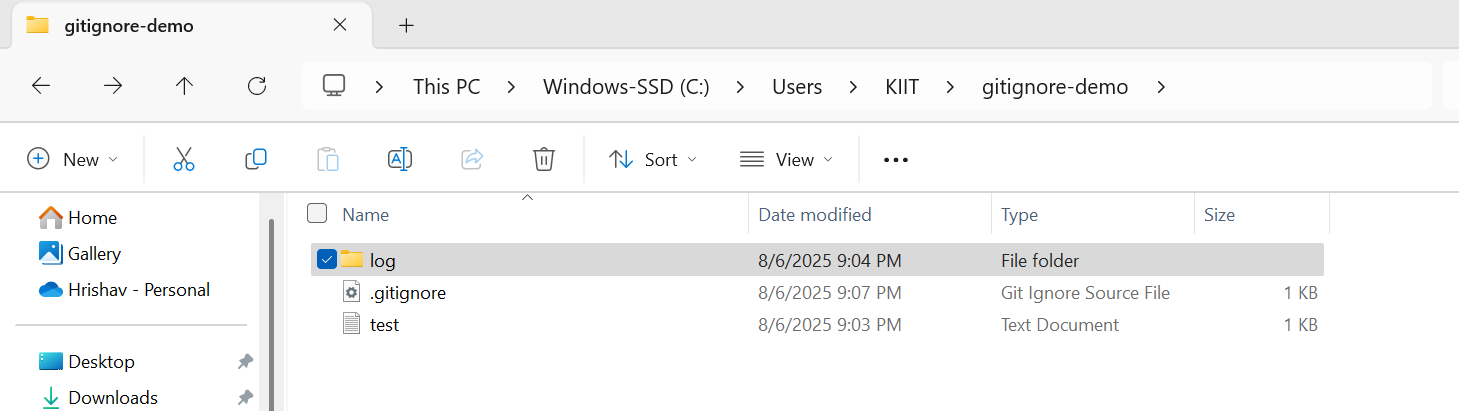
***Notepad will open and add the values which will have to ignore***



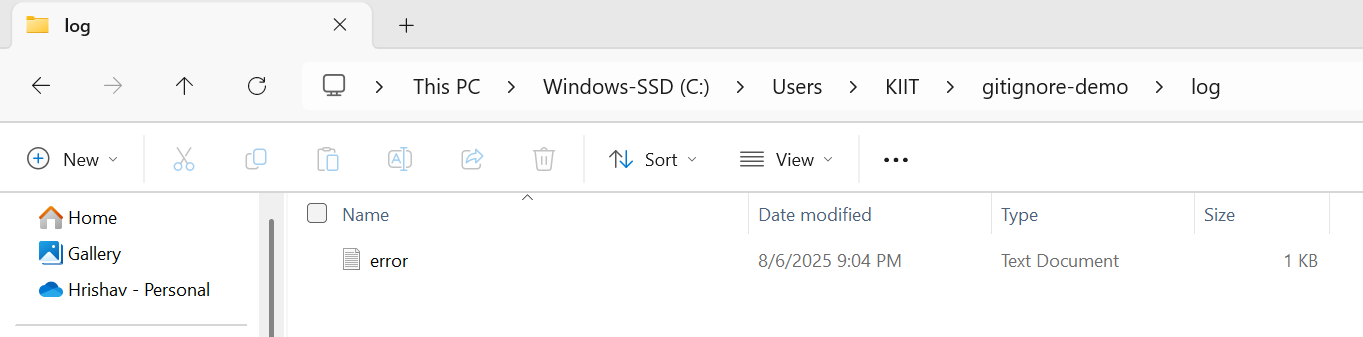
***Now Again Continue to get status updates and push the the code***



***Screenshots of files on local system***



***Inside the Log Files***



**Skill:-GIT**

**Filename:- 3. Git-HOL**

**Solution:-**

1. **Explain branching and merging**

**Answer:-**

**Branching:**

- A branch is a separate line of development.

- It lets you work on features, bug fixes, or experiments without affecting the main code (main or master branch).

**Merging:**

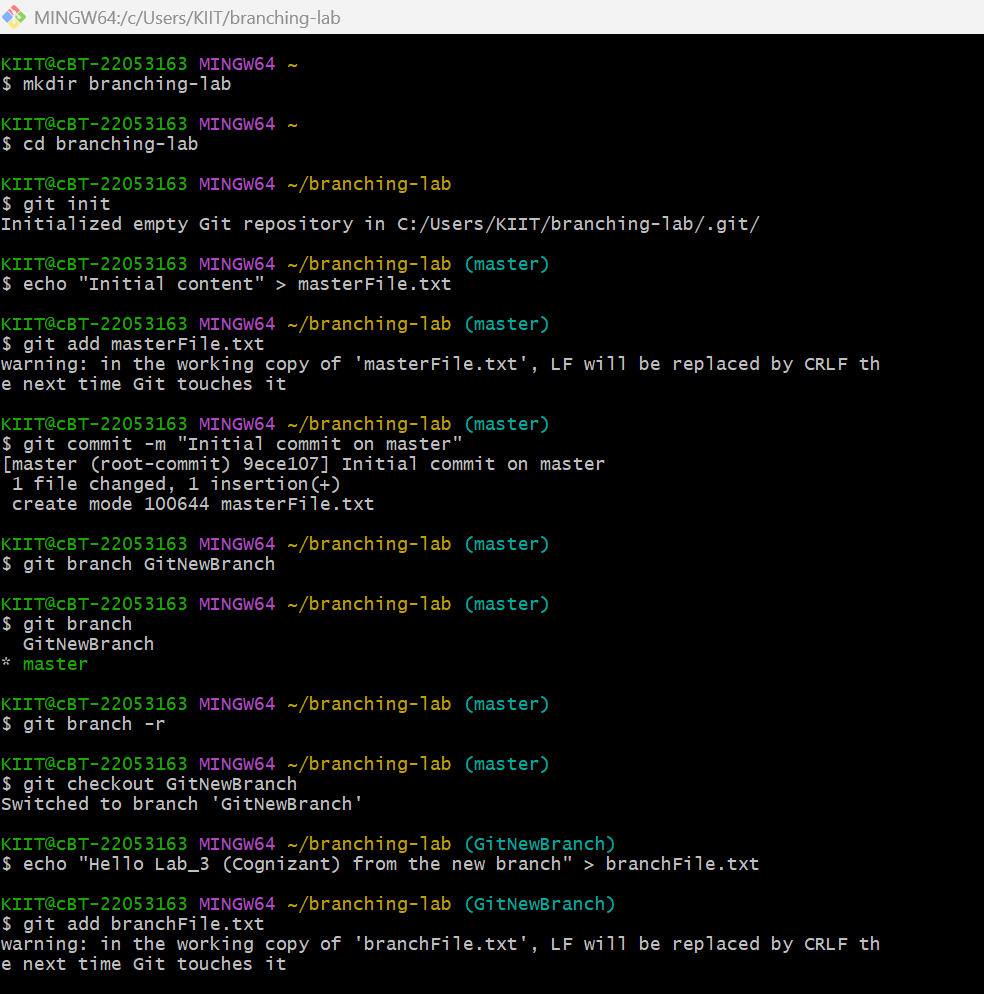
- **Merging** means combining the changes from one branch into another.

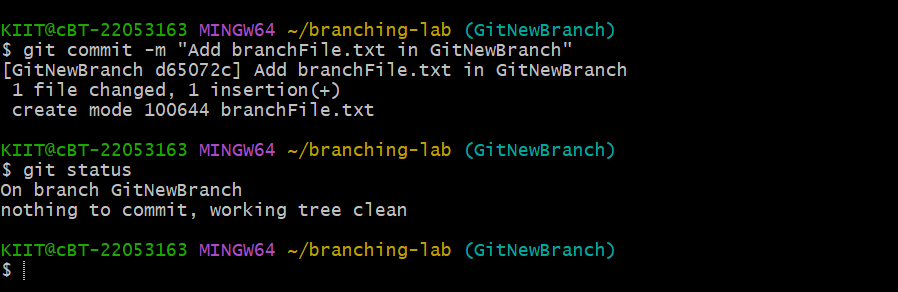
- Typically, when your work is done in a feature branch, you **merge it into main.**

**Branching:**

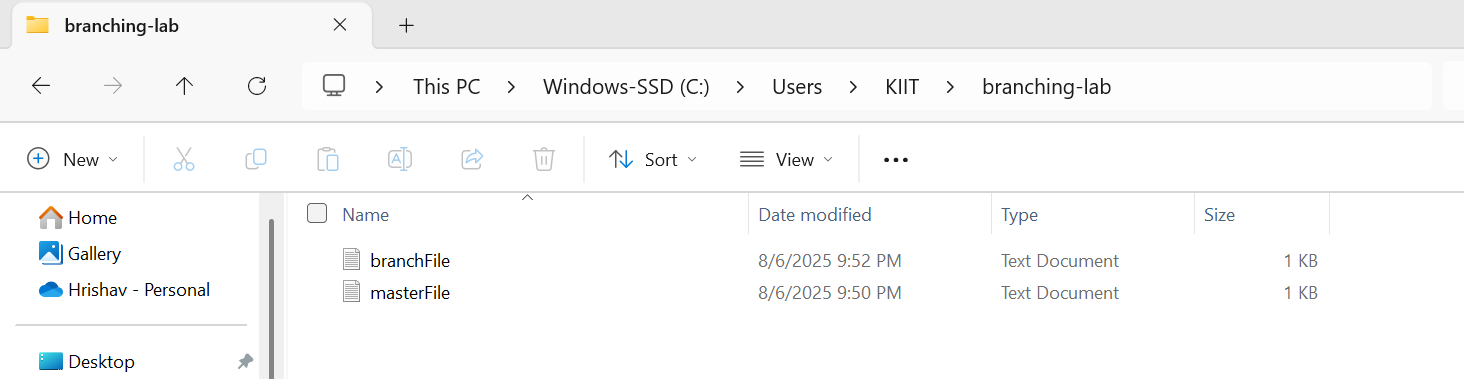
1. Create a new branch **“GitNewBranch”.**
2. List all the local and remote branches available in the current trunk. Observe the “\*” mark which denote the current pointing branch.
3. Switch to the newly created branch. Add some files to it with some contents.
4. Commit the changes to the branch.
5. Check the status with **“git status”** command.

***Output:-***





***Screenshots of files on local system***

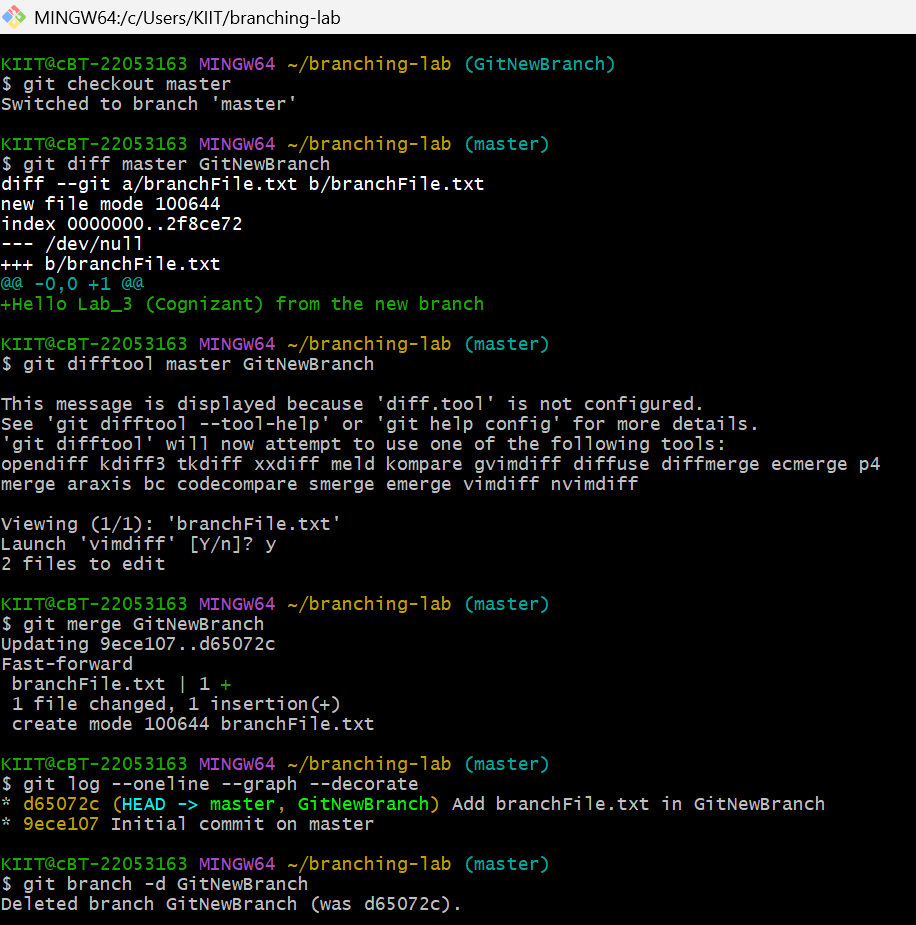


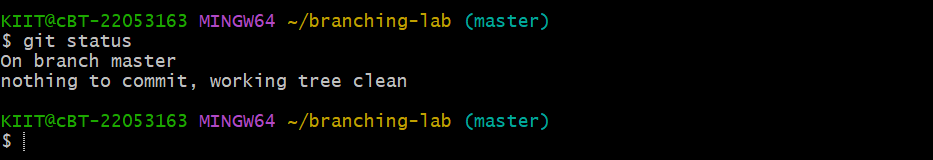
***Now we are working on Branching:-***

**Merging:**

1. Switch to the master
2. List out all the differences between trunk and branch. These provide the differences in command line interface.
3. List out all the visual differences between master and branch using **P4Merge tool**.
4. Merge the source branch to the trunk.
5. Observe the logging after merging using **“git log –oneline –graph –decorate”**
6. Delete the branch after merging with the trunk and observe the git status.

***Output:-***





**Skill:-GIT**

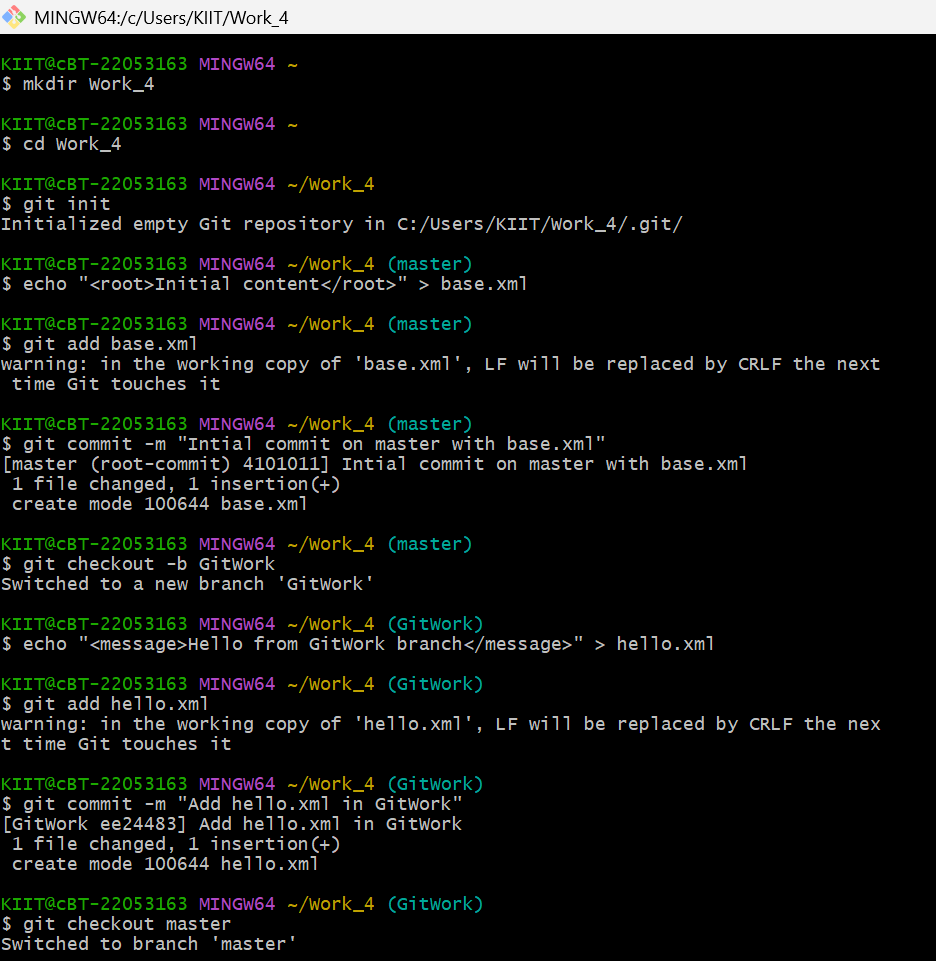
**Filename:- 4. Git-HOL**

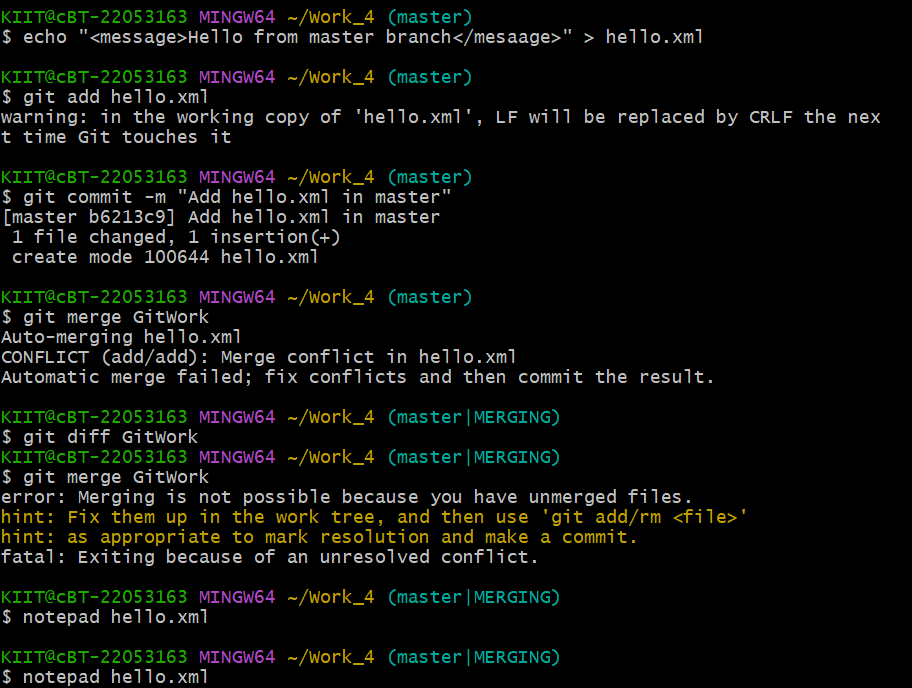
**Solution:-**

Please follow the instructions to complete the hands-on. Each instruction expect a command for the Git Bash.

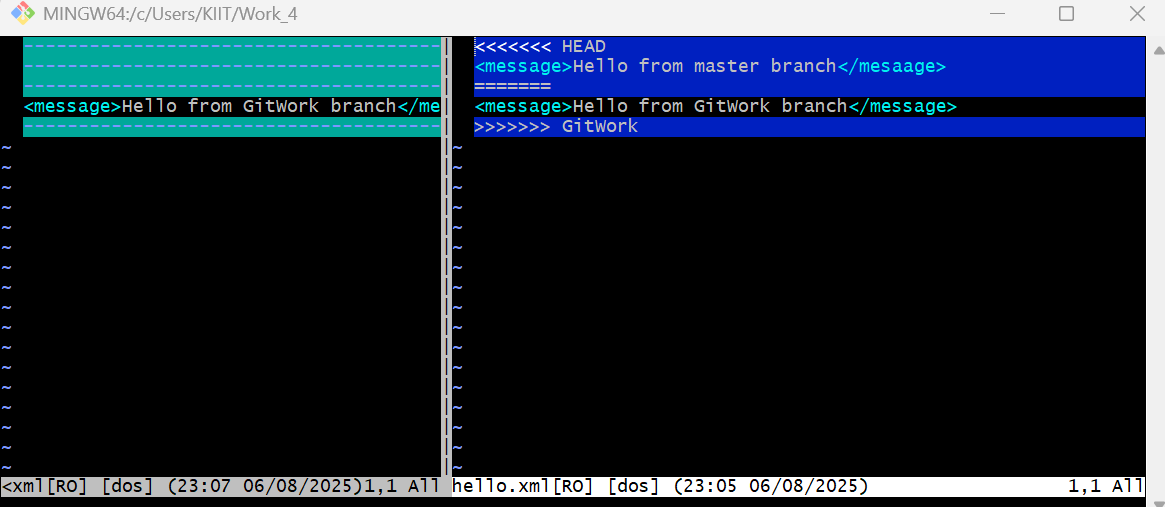
1. Verify if master is in clean state.
2. Create a branch **“GitWork”.** Add a file “hello.xml”.
3. Update the content of “hello.xml” and observe the status
4. Commit the changes to reflect in the branch
5. Switch to master.
6. Add a file **“hello.xml”** to the master and add some different content than previous.
7. Commit the changes to the master
8. Observe the log by executing **“git log –oneline –graph –decorate –all”**
9. Check the differences with Git diff tool
10. For better visualization, use P4Merge tool to list out all the differences between master and branch
11. Merge the bran to the master
12. Observe the git mark up.
13. Use 3-way merge tool to resolve the conflict

***Output:-***

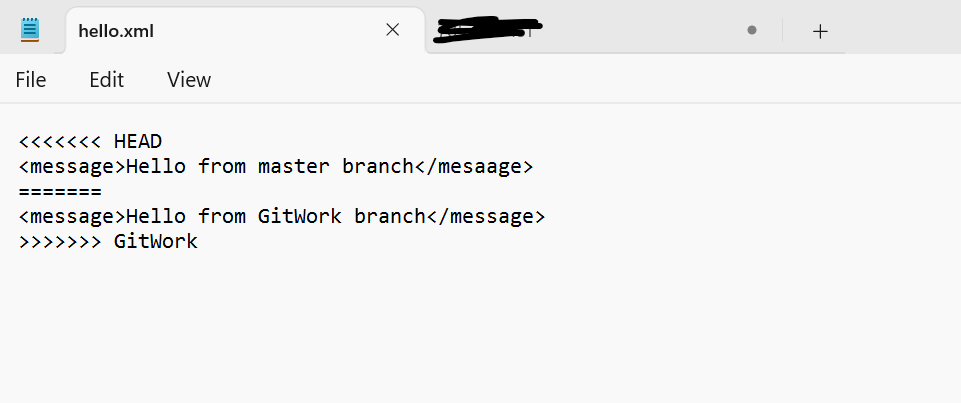


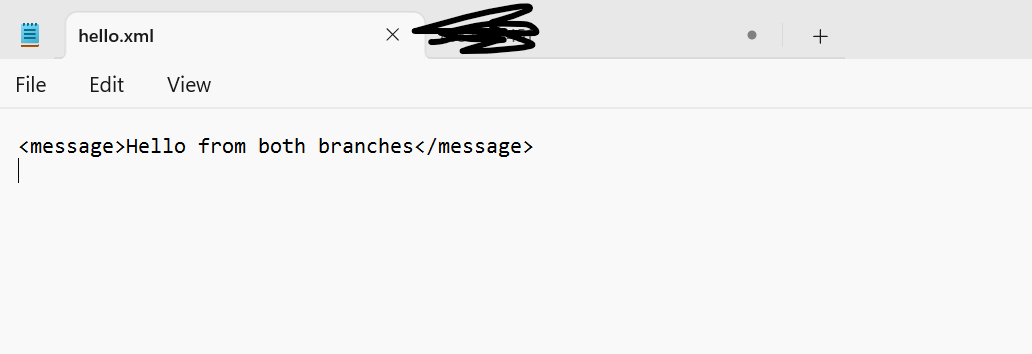


***Visual diff with P4 Merge***



***Notepad View the error log***

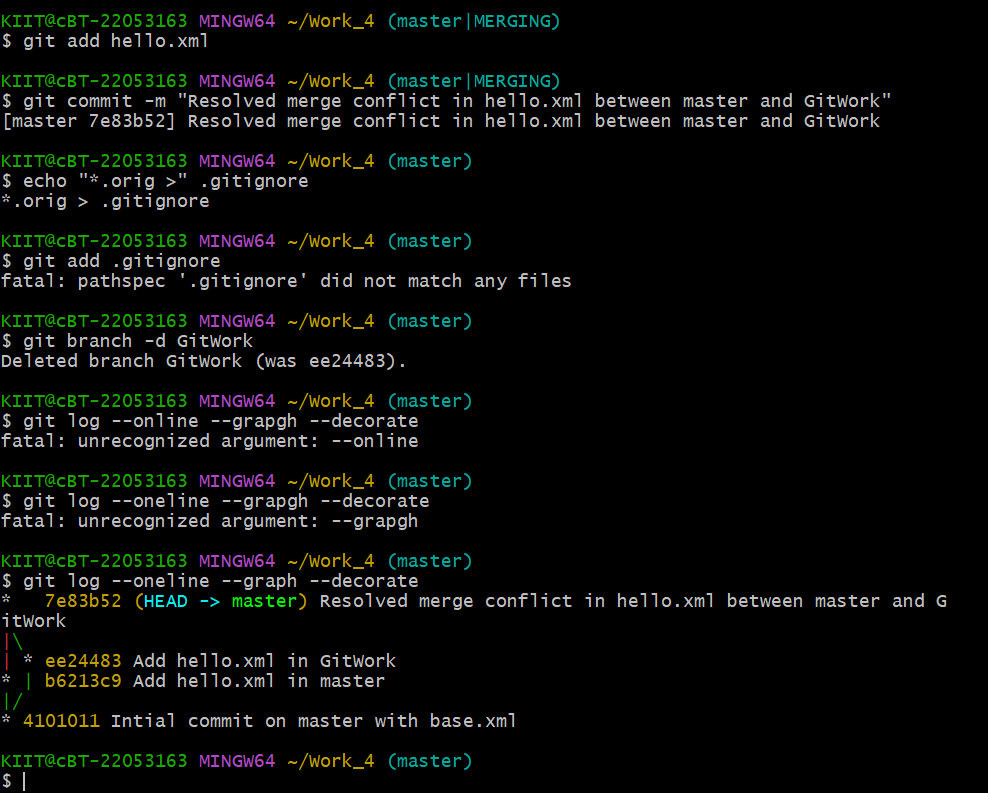


***After Updating Notepad***  
  


***Now,***

1. Commit the changes to the master, once done with conflict
2. Observe the git status and add backup file to the .gitignore file.
3. Commit the changes to the .gitignore
4. List out all the available branches
5. Delete the branch, which merge to master.
6. Observe the log by executing “git log –oneline –graph –decorate”

***Now the conflict is resolved and now commit***



**Skill:-GIT**

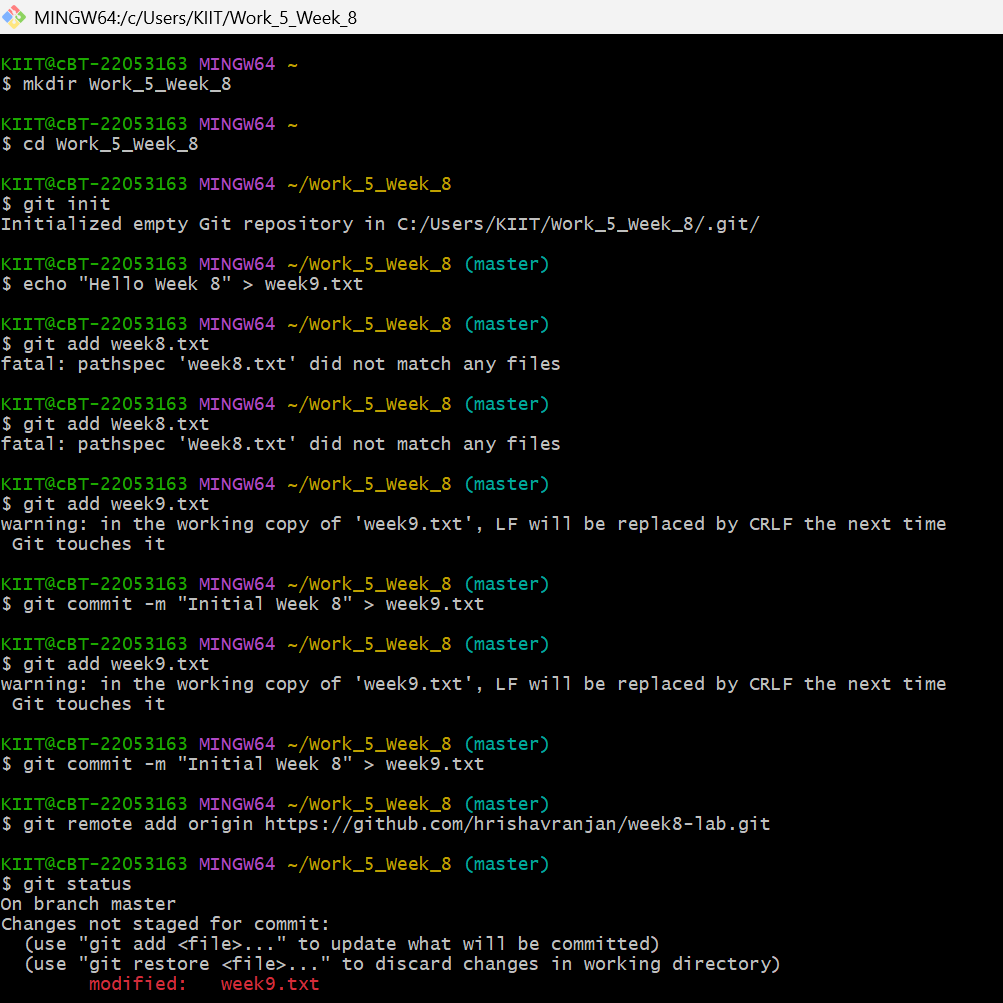
**Filename:- 5. Git-HOL**

**Solution:-**

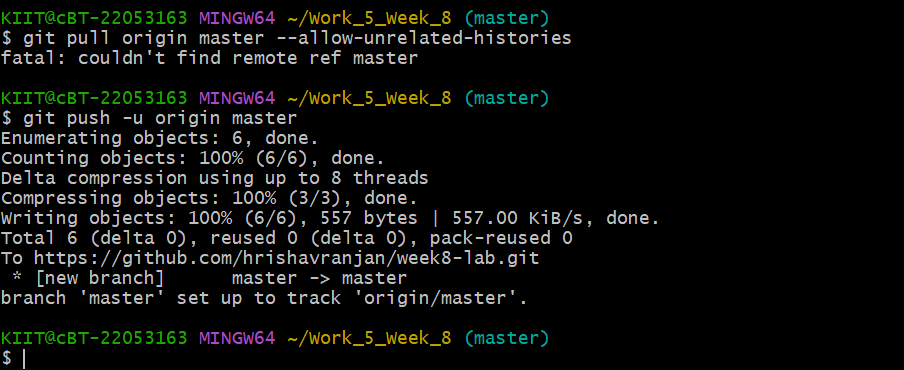
Please follow the instructions to complete the hands-on. Each instruction expects a command for the Git Bash.

1. Verify if master is in clean state.
2. List out all the available branches.
3. Pull the remote git repository to the master
4. Push the changes, which are pending from **“Git-T03-HOL\_002”** to the remote repository.
5. Observe if the changes are reflected in the remote repository.

***Output:-***

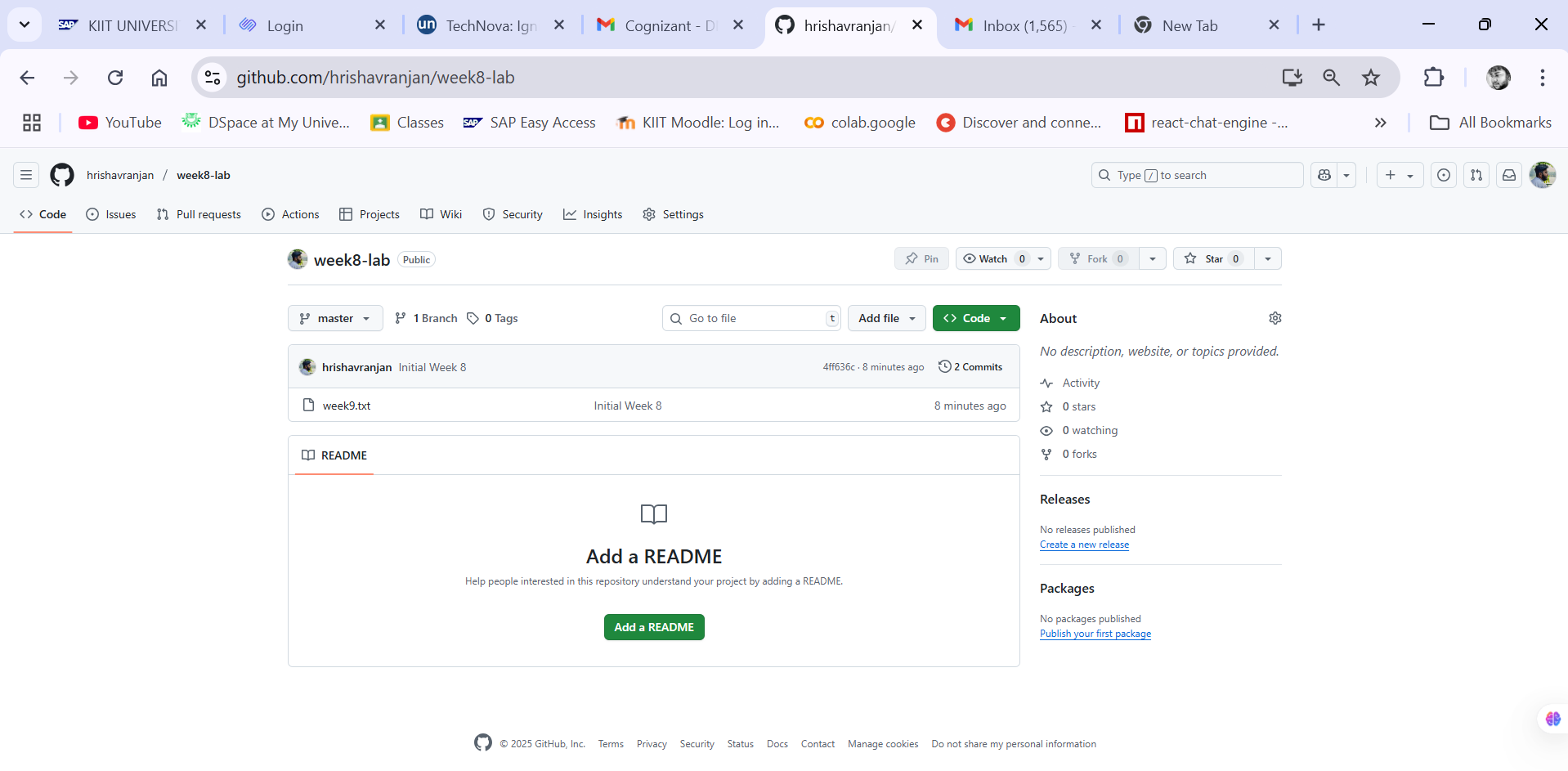






***Now Open Github on browser of Your Laptop checked Github***

***Results:-***



***Close View***

