**WEEK 8**

**1.Git-HOL**

Objectives Completed:

Step 1: Git Configuration

* Checked Git version: git --version
* Set username and email:

git config --global user.name "Your Name"

git config --global user.email "your@email.com"

* Verified config: git config --global --list

Step 2: Notepad++ Integration

* Added Notepad++ to system path
* Ran notepad++ in Git Bash — it opened
* *(Optional: You can now set it as Git default editor if needed)*

Step 3: Create & Push Git Project

* Created GitDemo project in GitLab
* Initialized local Git repo: git init
* Created and added files: git add .
* Committed changes: git commit -m "initial commit"
* Linked remote repo:

git remote add origin https://gitlab.com/your-username/GitDemo.git

* Pushed code:

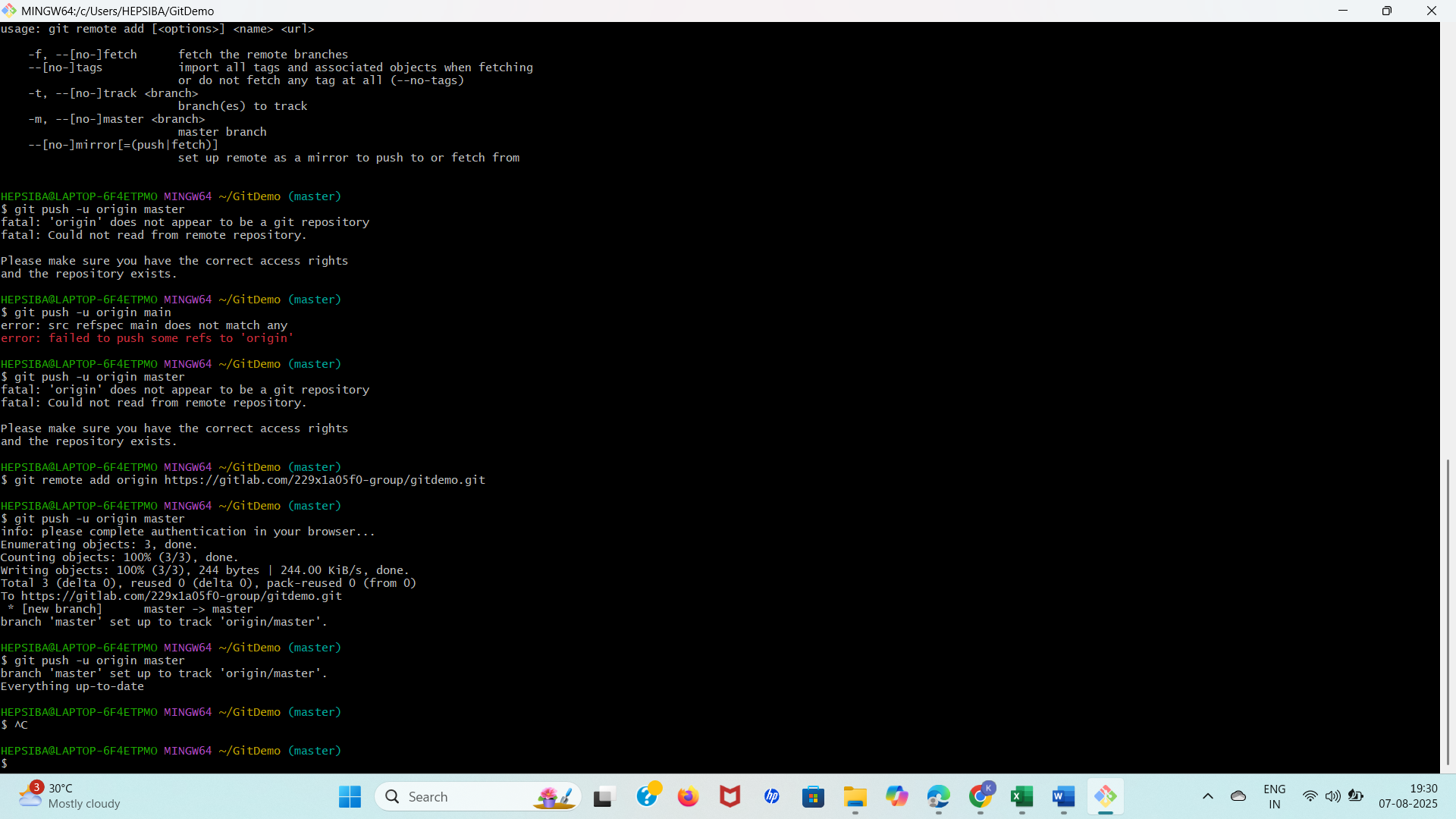
git push -u origin master

* **Output:**

branch 'master' set up to track 'origin/master'.

Everything up-to-date

**OUTPUT**

****

**2.Git-HOL**

1. Create and navigate into folder

$ mkdir GitIgnoreDemo

$ cd GitIgnoreDemo

2. Initialize Git

$ git init

**Output:**

Initialized empty Git repository in C:/Users/YourName/GitIgnoreDemo/.git/

3. Create .log file and log/ folder

$ echo "this is a log file" > test.log

$ mkdir log

$ echo "this is a log inside folder" > log/info.txt

4. Open .gitignore and add rules

$ notepad .gitignore

\*.log

log/

5. Check Git status

$ git status

Output:

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

.gitignore

nothing added to commit but untracked files present (use "git add" to track)

6. Add and commit .gitignore

$ git add .gitignore

$ git commit -m "Add .gitignore to ignore .log files and log folder"

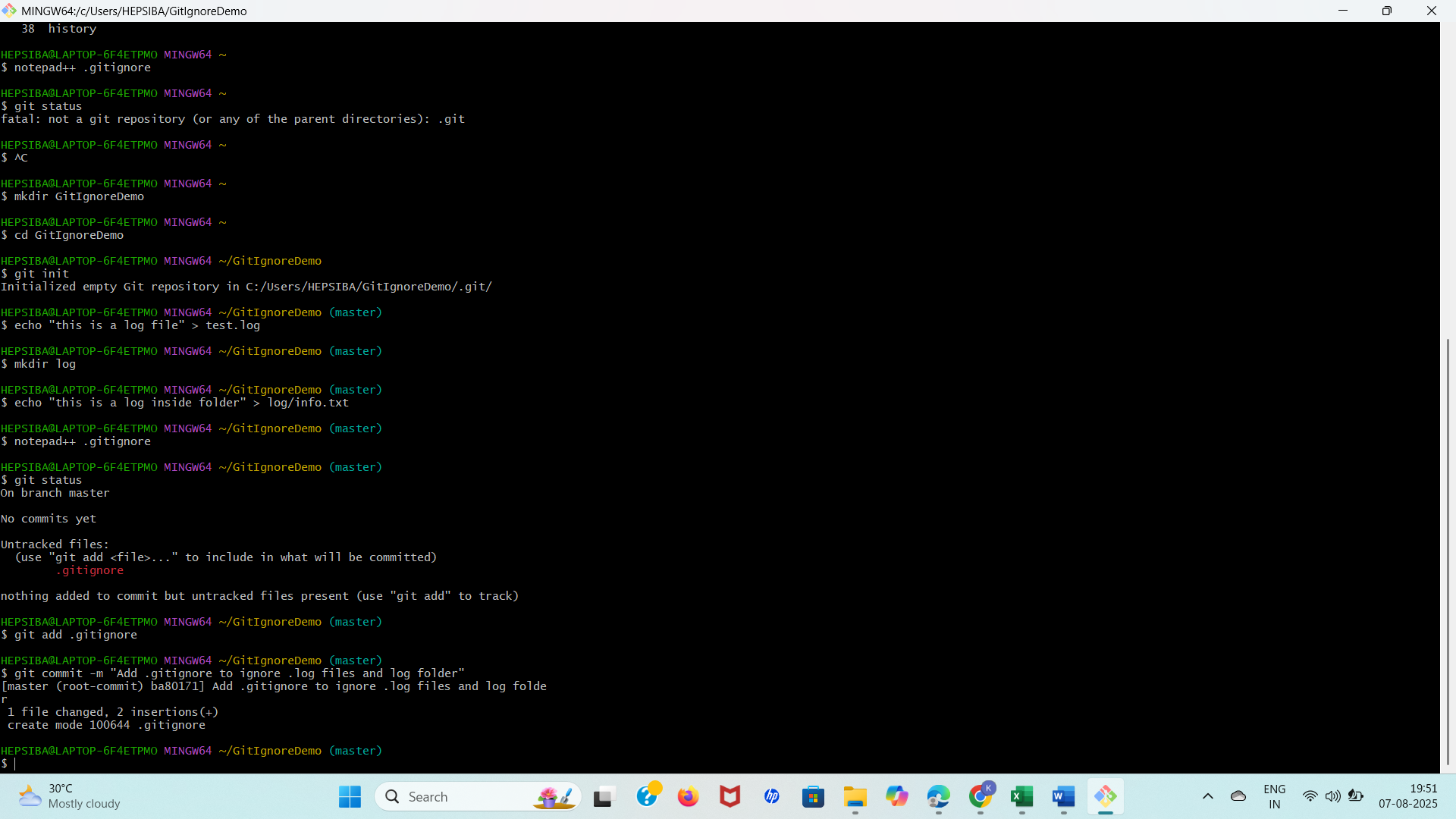
Output:

[master (root-commit) abc1234] Add .ignore to ignore .log files and log folder

1 file changed, 2 insertions(+)

create mode 100644 .ignore

**OUTPUT**



**3.Git-HOL**

**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.

**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.

$ git branch GitNewBranch

fatal: a branch named 'GitNewBranch' already exists

$ git branch

GitNewBranch

\* master

$ git checkout GitNewBranch

Switched to branch 'GitNewBranch'

$ echo "This is work done in GitNewBranch" > branchwork.txt

$ git add branchwork.txt

warning: in the working copy of 'branchwork.txt', LF will be replaced by CRLF the next time Git touches it

$ git commit -m "Added branchwork.txt in GitNewBranch"

[GitNewBranch 1d6079a] Added branchwork.txt in GitNewBranch

1 file changed, 1 insertion(+)

create mode 100644 branchwork.txt

$ git checkout master

Switched to branch 'master'

$ git merge GitNewBranch

Updating ba80171..1d6079a

Fast-forward

branchwork.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 branchwork.txt

$ git branch -d GitNewBranch

Deleted branch GitNewBranch (was 1d6079a).

$ git status

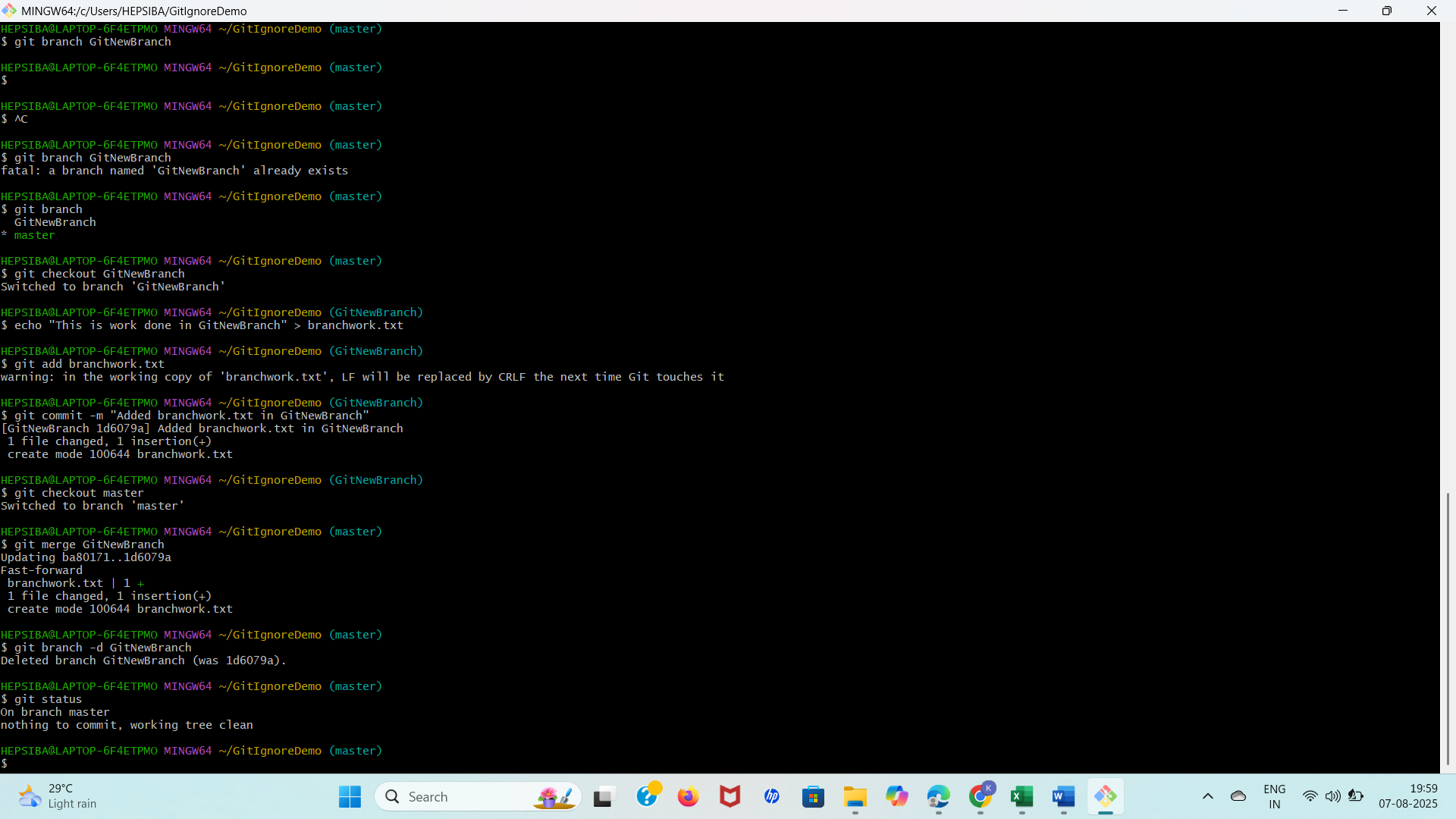
On branch master

nothing to commit, working tree clean

$ git log --oneline --graph --decorate

\* 1d6079a (HEAD -> master) Added branchwork.txt in GitNewBranch

\* ba80171 Initial commit (example)



**4.Git-HOL**

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
14. Commit the changes to the master, once done with conflict
15. Observe the git status and add backup file to the .gitignore file.
16. Commit the changes to the .gitignore
17. List out all the available branches
18. Delete the branch, which merge to master.
19. Observe the log by executing **“git log –oneline –graph –decorate”**

**1. Ensure master is in a clean state**

git checkout master

git status

**2. Create a branch GitWork and add hello.xml**

git checkout -b GitWork

echo "<msg>Hello from GitWork branch</msg>" > hello.xml

git status

**3. Update hello.xml**

echo "<msg>Updated in GitWork branch</msg>" > hello.xml

git status

**4. Commit the changes**

git add hello.xml

git commit -m "Added hello.xml with GitWork content"

**5. Switch to master and make conflicting change**

git checkout master

echo "<msg>Hello from master branch</msg>" > hello.xml

git add hello.xml

git commit -m "Added hello.xml with master content"

**6. View logs and diffs**

git log --oneline --graph --decorate --all

git diff master GitWork

**8. Attempt to merge GitWork into master**

git merge GitWork

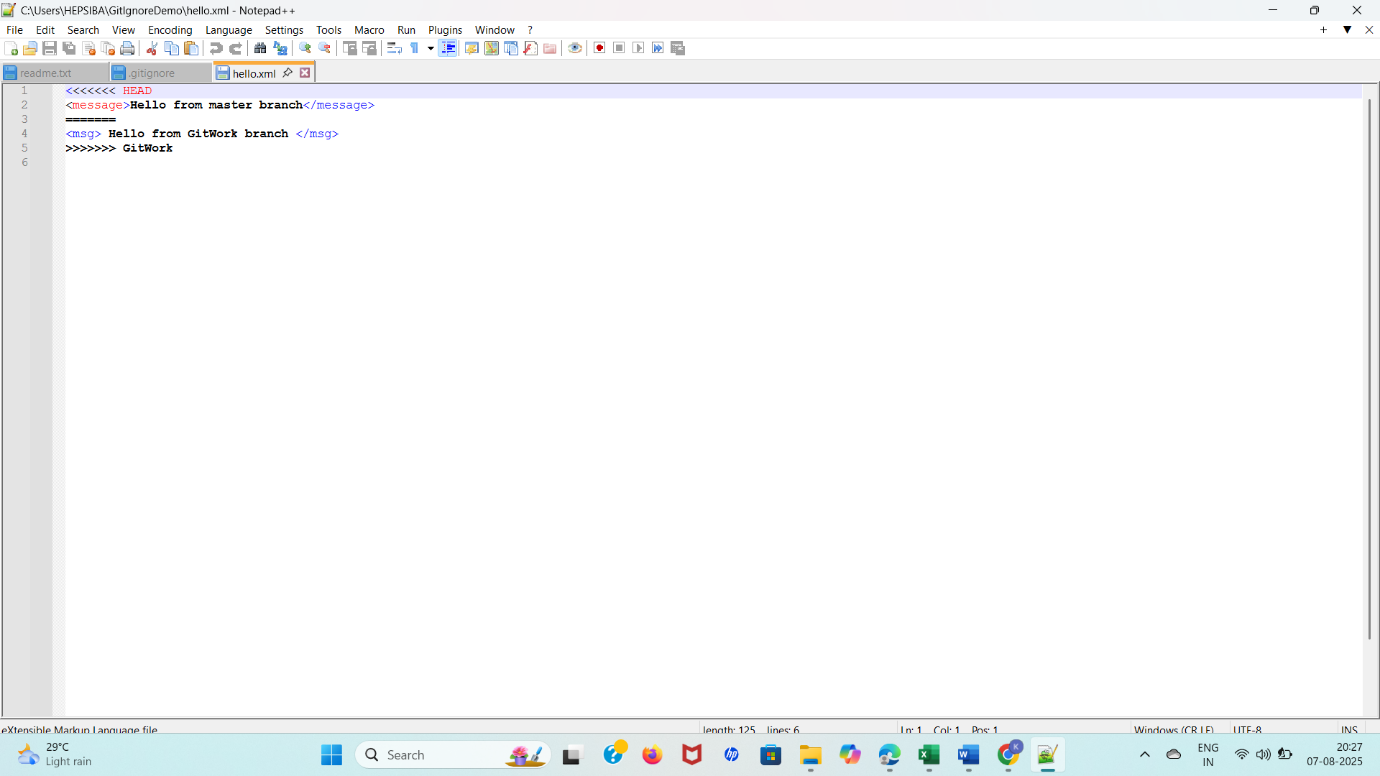
<<<<<<< HEAD

<msg>Hello from master branch</msg>

=======

<msg>Updated in GitWork branch</msg>

>>>>>>> GitWork



**5.Git-HOL**

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.

cd ~/GitIgnoreDemo

git checkout master

# Check clean working directory

git status

# Create and switch to a new branch

git checkout -b GitWork

# Add a file in GitWork branch

echo "<message> Hello from GitWork branch </message>" > hello.xml

git add hello.xml

git commit -m "Added hello.xml in GitWork"

# Switch back to master

git checkout master

# Add same file (with different content) in master

echo "<message> Hello from master branch </message>" > hello.xml

git add hello.xml

git commit -m "Added hello.xml in master with different content"

# View git log

git log --oneline --graph --decorate –all

**OUTPUT**

