**2. Git – HOL**

**Objectives:**

* Explain git ignore.
* Explain how to ignore unwanted files using git ignore.
* Implement git ignore command to ignore unwanted files and folders.

**Prerequisites:**

* Setting up Git environment (I've completed this in the previous lab!).
* Integrate Notepad++ as a default editor (I've completed this!).
* A Git repository in the local system and a remote repository in GitLab (I've completed this with your GitDemo project!).

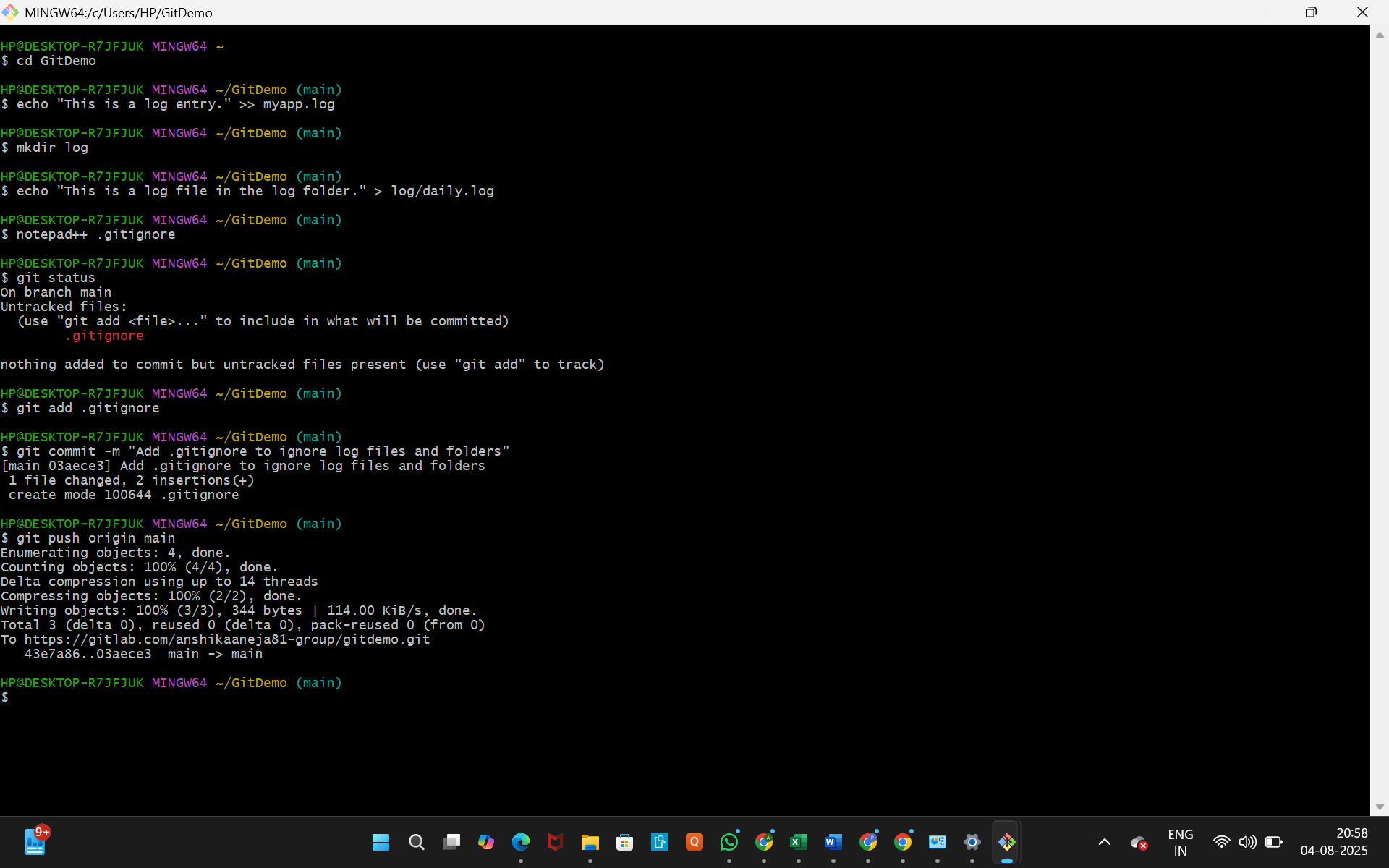
**The main task is described in the last paragraph. Let's break it into actionable steps:**

**Step 1: Understand git ignore (Conceptual)**

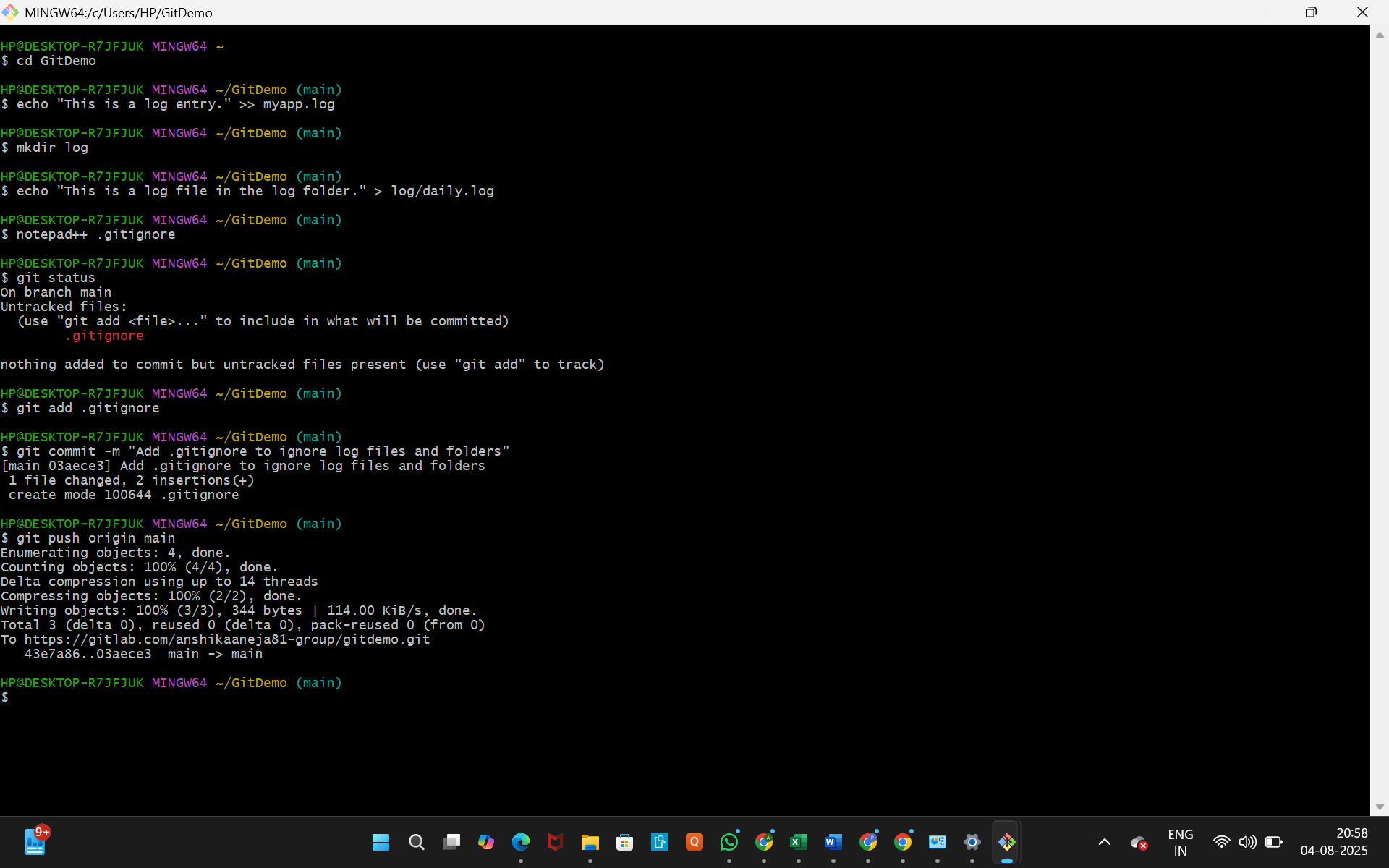
* **What is git ignore?**
  + git ignore refers to a special file named .gitignore that you place in your Git repository.
  + Its purpose is to tell Git which files and directories to *ignore* and *not* track. This is crucial for avoiding committing temporary files, build artifacts, personal configuration files, operating system files (like .DS\_Store on macOS or Thumbs.db on Windows), log files, and other files that are not part of your project's source code or are generated during its execution.
  + When a file or folder is listed in .gitignore, Git will simply disregard it when checking for changes (git status) and will not include it in commits.

**Step 2: Create a .log file in the working directory.**

* **Action:** Create a new file in your GitDemo working directory.
* **How to do it:**
  1. Open your Git Bash shell.
  2. Navigate to your GitDemo project directory (e.g., cd D:/Development\_Avec/GitDemo or simply cd GitDemo if you're in the parent directory).



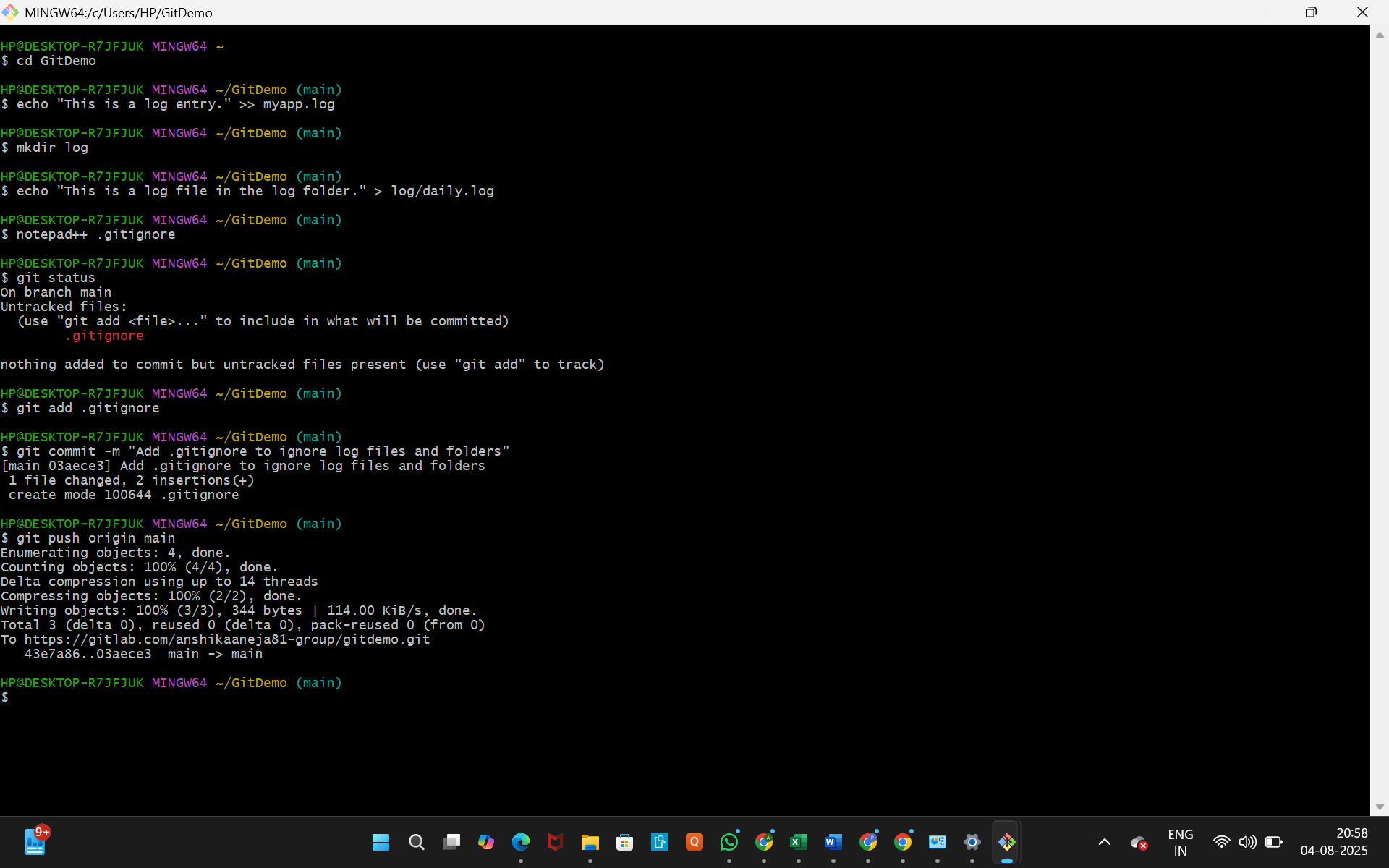
* 1. Execute the following command to create a sample log file:



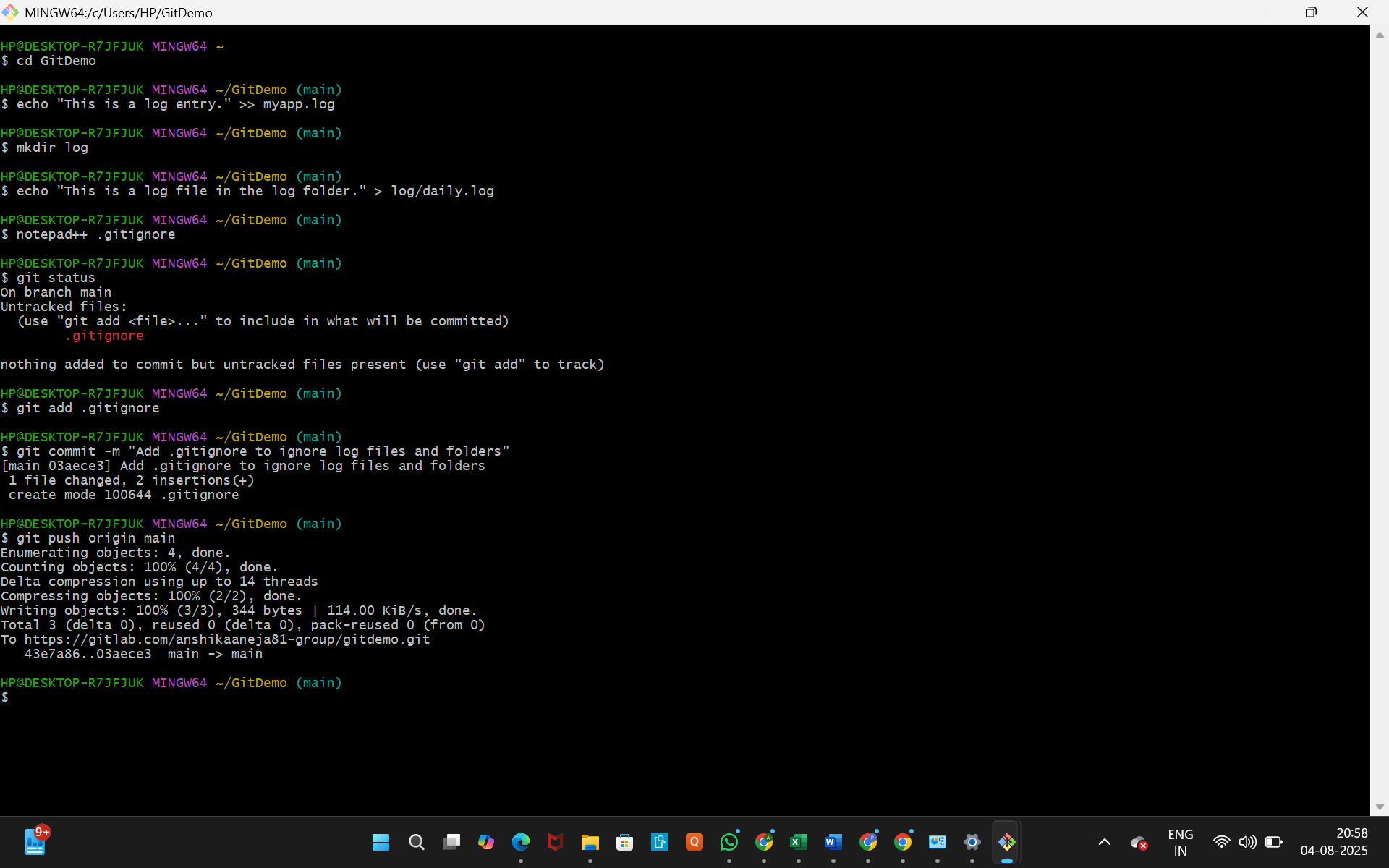
(You can name it anything, like error.log, debug.log, etc., as long as it has a .log extension).

**Step 3: Create a log folder in the working directory.**

* **Action:** Create a new directory (folder) in your GitDemo working directory.
* **How to do it:**
  1. In your Git Bash shell (still in the GitDemo directory), execute the following command:

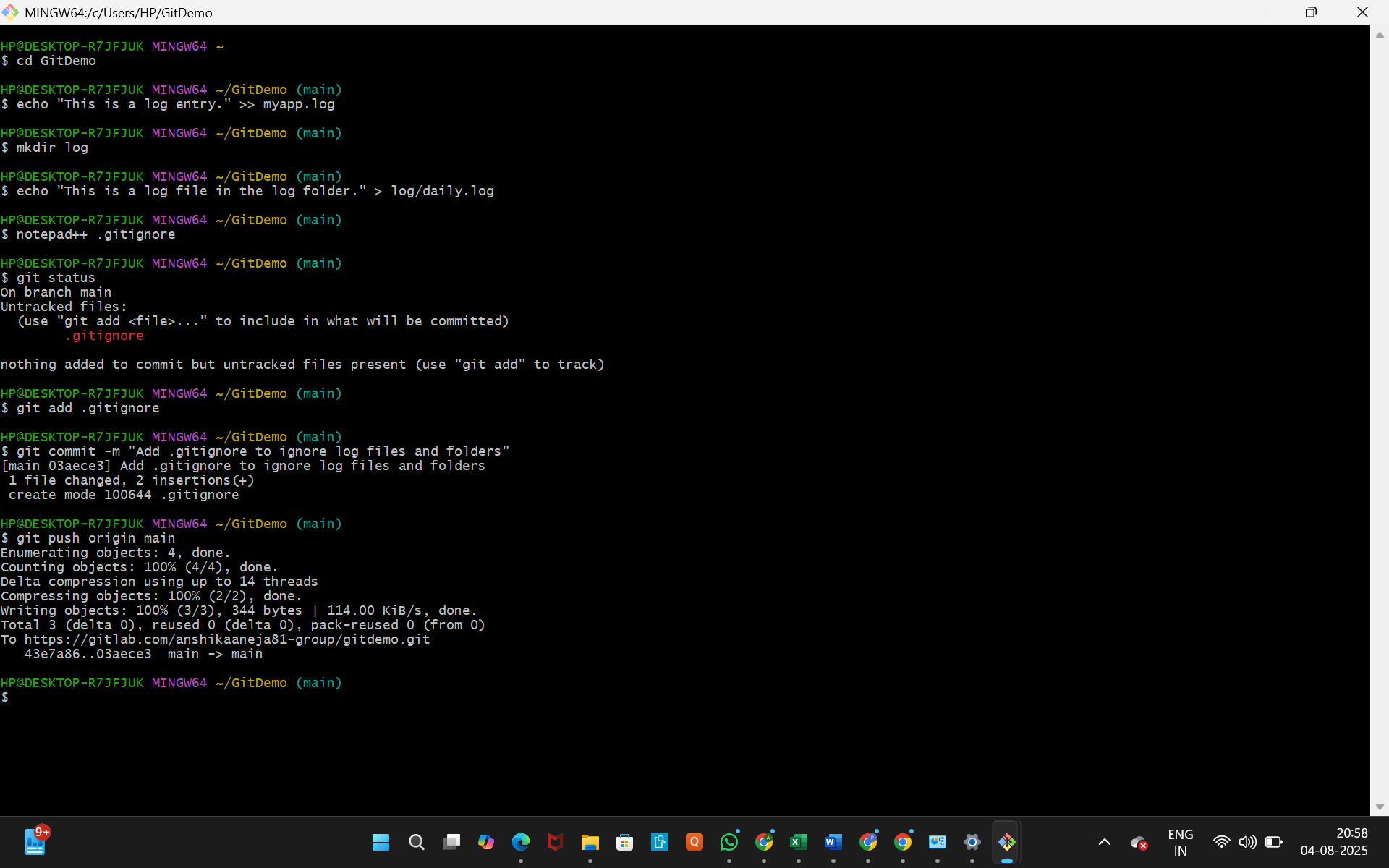


* 1. (Optional, but good for demonstrating ignore): Create a dummy file *inside* this new log folder.



**Step 4: Update the .gitignore file to ignore .log file extensions and log folders.**

* **Action:** Create or modify the .gitignore file and add rules to ignore the specified files/folders.
* **How to do it:**
  1. In your Git Bash shell (still in the GitDemo directory), execute the command to open/create .gitignore using your default editor (Notepad++):



(If the file doesn't exist, Notepad++ might ask if you want to create it – click "Yes").

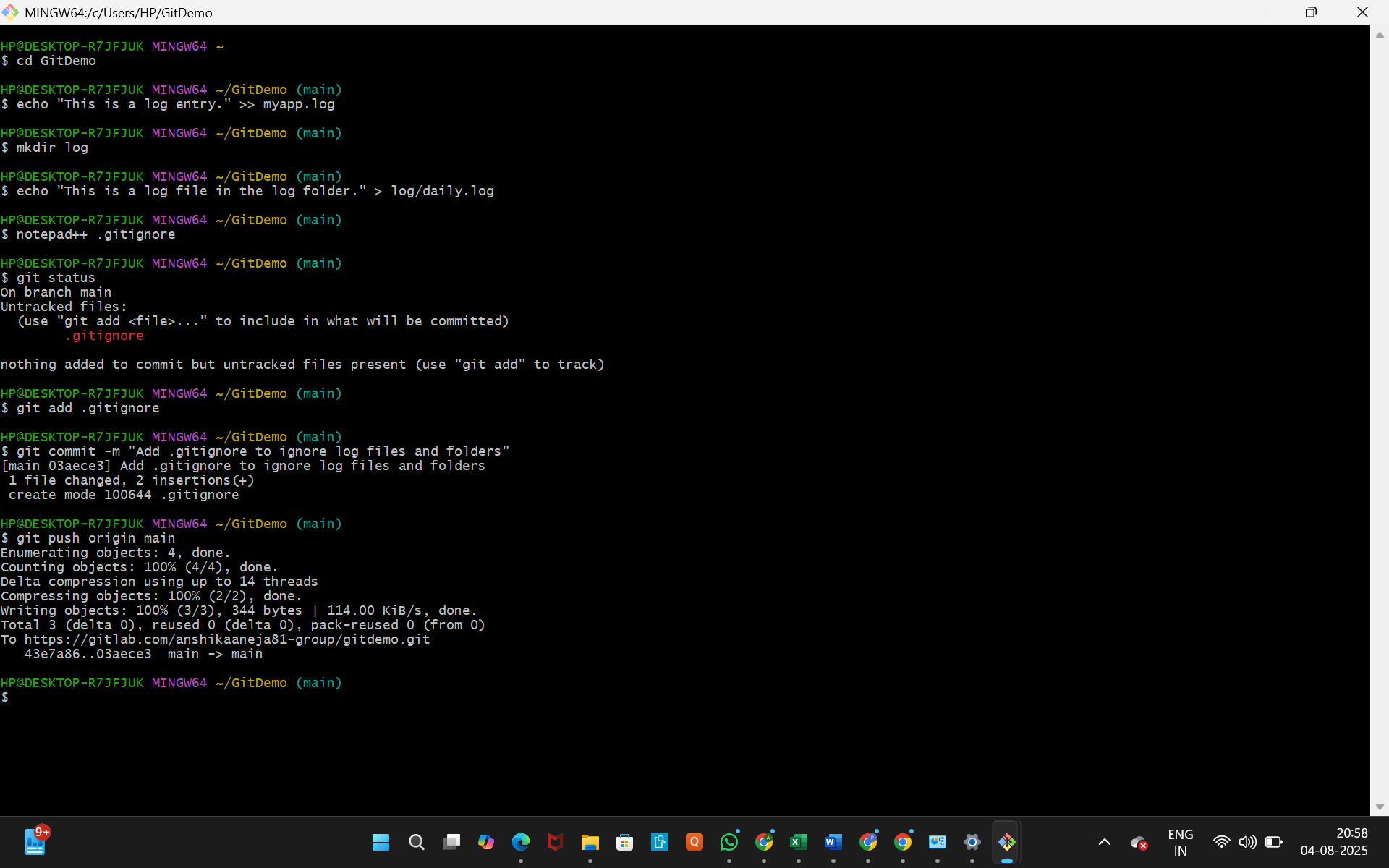
* 1. In the Notepad++ window that opens for .gitignore, add the following two lines:



* + - **\*.log**: This rule tells Git to ignore any file that ends with .log (e.g., myapp.log, error.log, server.log).
    - **log/**: This rule tells Git to ignore any directory named log (and all its contents) anywhere in the repository. The trailing slash / is important to specify it as a directory.
  1. Save the .gitignore file (Ctrl + S) and close Notepad++.

**Step 5: Verify if git status reflects the same about working directory, local repository, and Git repository (i.e., the .log file and log folder are now ignored).**

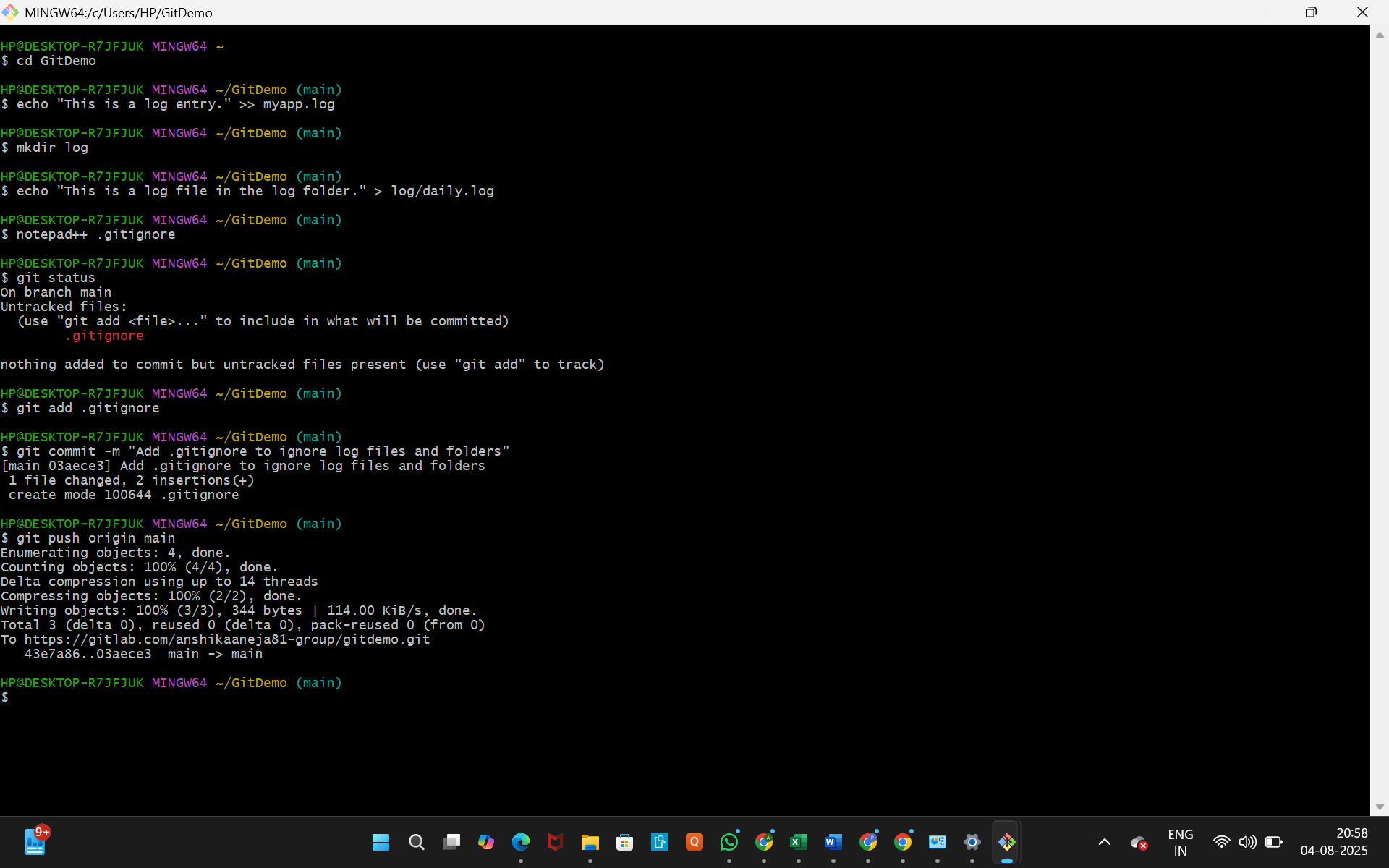
* **Action:** Use git status to see the current state of your repository after adding the ignore rules.
* **How to do it:**
  1. In your Git Bash shell, execute the command:



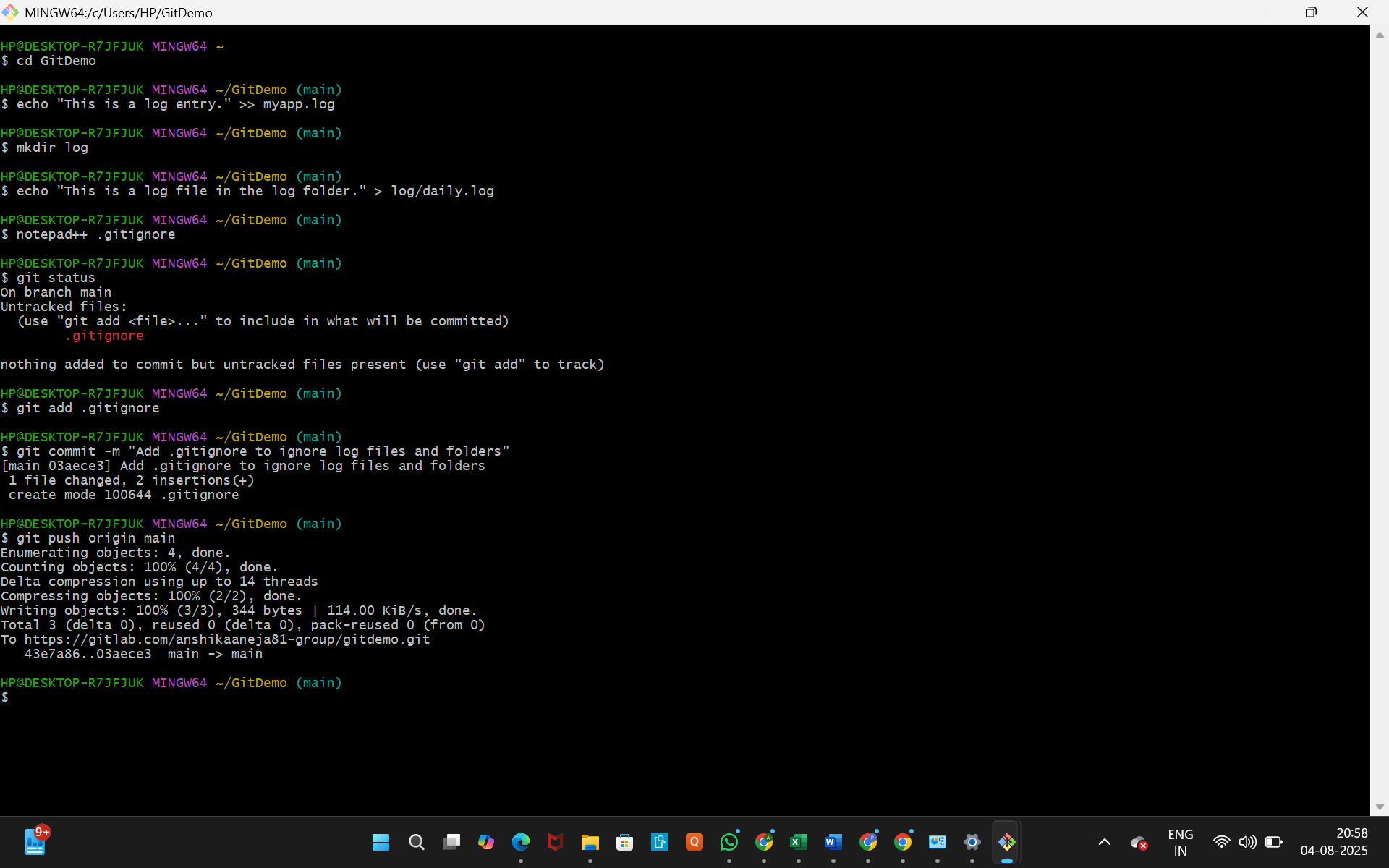
* **What to expect:**
  1. You should NOT see myapp.log or the log/ directory (and its contents like log/daily.log) listed as "Untracked files" anymore. This means Git is successfully ignoring them.
  2. You WILL see .gitignore listed as an "Untracked file" or "Changes to be committed". This is because the .gitignore file itself is a part of your project's configuration and *should* be committed to the repository so that everyone working on the project uses the same ignore rules.

**Step 6: Commit the .gitignore file (Implicit Next Step for Good Practice).**

* **Action:** Stage and commit the .gitignore file**.**
* **How to do it:**
  1. In your Git Bash shell, execute:



* 1. Then, execute:

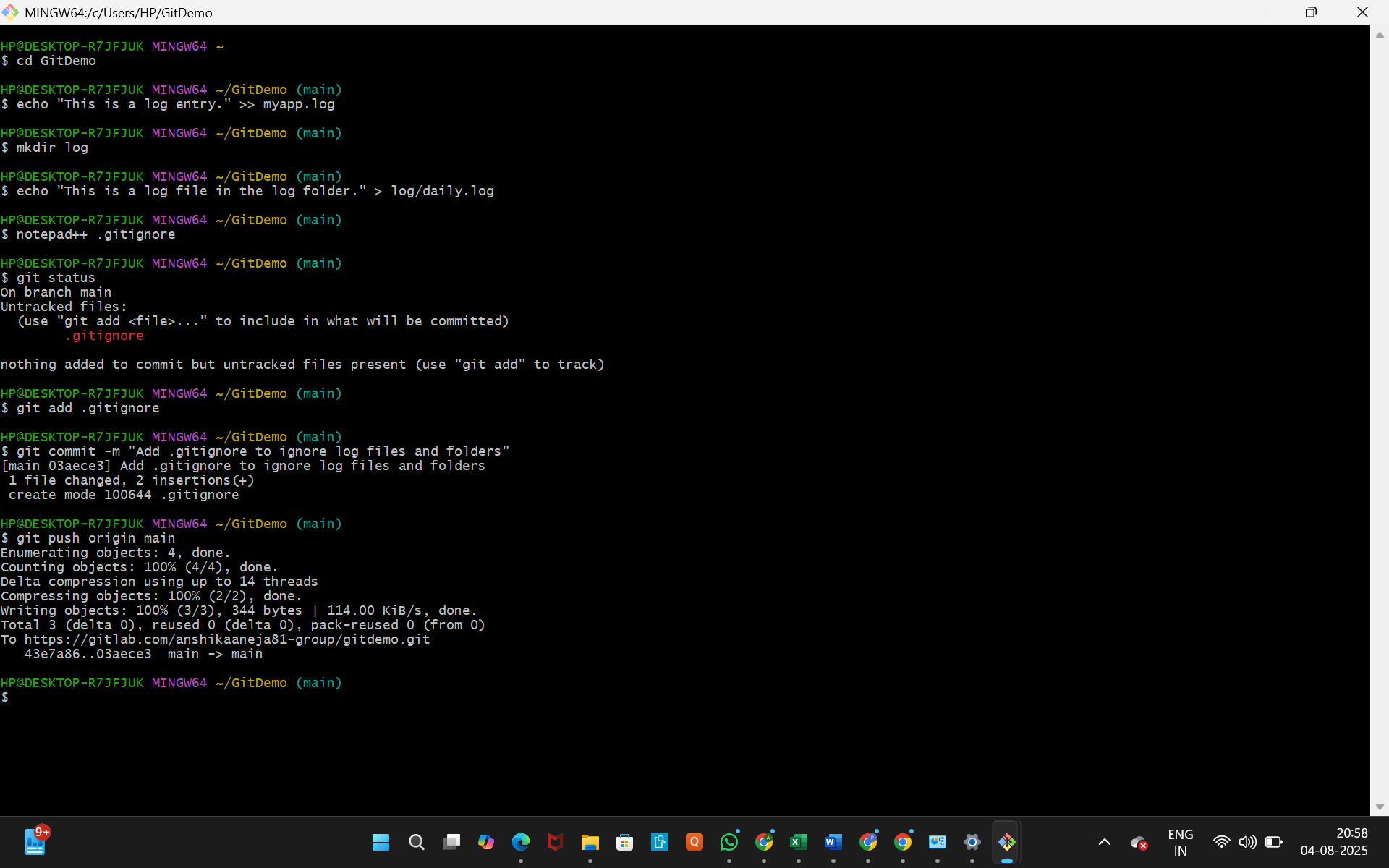


(The -m flag allows you to add a commit message directly without opening the editor).

* **What to expect:** Git will record your .gitignore file in your repository.

**Step 7: Push the changes to the remote GitLab repository (Implicit Next Step for Good Practice).**

* **Action:** Push your committed .gitignore file to GitLab.
* **How to do it:**
  1. In your Git Bash shell, execute:



* **What to expect:** Your .gitignore file will now be visible on your GitLab repository, ensuring that anyone who clones or pulls from your repo will automatically ignore the same files.