## Exercise 2 Git-HOL

## 1. Introduction

The `.gitignore` file tells Git which files or folders it should not track.  
  
This is important for ignoring:  
- Temporary files (like logs, caches, or OS-generated files).  
- Files containing sensitive information (like API keys or passwords).  
- Build output or compiled files that can be regenerated.  
  
Anything listed in `.gitignore` will be ignored by Git when checking for changes.  
However, note: If a file is already being tracked by Git, adding it to `.gitignore` will not remove it — you have to untrack it first.  
  
Objectives:  
- Explain `.gitignore` and its purpose.  
- Create ignore rules to skip certain files and folders.  
- Verify that Git respects these rules.

## Step 1: Prepare the Working Directory

We will work inside our existing GitDemo repository from the previous lab.  
  
1. Move into your repository:  
 cd GitDemo  
 # Make sure you are in the correct folder for your project.  
  
2. Check current Git status:  
 git status  
 # This confirms that your working directory is clean before we start.  
 # If there are pending changes, either commit or stash them.

## Step 2: Create Files and Folders to be Ignored

We’ll create:  
- A `.log` file (e.g., `app.log`).  
- A folder named `log` with a file inside (`debug.log`).  
  
1. Create the `app.log` file:  
 touch app.log  
 # This simulates a log file generated by an application.  
  
2. Create the `log` directory and a file inside it:  
 mkdir log  
 touch log/debug.log  
 # This simulates an application logging system that stores multiple files inside a folder.  
  
3. Check status to see these as untracked files:  
 git status  
 # Output: You should see `app.log` and `log/` listed under 'Untracked files'.

## Step 3: Create and Configure the .gitignore File

The `.gitignore` file contains rules that tell Git what to skip.  
  
We’ll add:  
- `\*.log` → Ignore any file ending with `.log`.  
- `log/` → Ignore the entire `log` folder and its contents.  
  
1. Create the `.gitignore` file:  
 touch .gitignore  
 # This will store our ignore patterns.  
  
2. Add ignore rules to `.gitignore`:  
 echo "\*.log" >> .gitignore  
 echo "log/" >> .gitignore  
 # The first rule ignores any file with the `.log` extension.  
 # The second rule ignores the entire `log` folder.  
  
3. Verify `.gitignore` contents:  
 cat .gitignore  
 # Output should be:  
 # \*.log  
 # log/

## Step 4: Verify Ignoring is Working

Check Git status again to see if `.log` files and the `log/` folder are gone from the untracked list.  
  
git status  
# Now you should only see `.gitignore` as untracked.  
# This means Git is ignoring `app.log` and `log/`.

## Step 5: Commit the .gitignore File

It’s best practice to commit `.gitignore` so others working on the same repository use the same rules.  
  
1. Add `.gitignore` to staging:  
 git add .gitignore  
 # Or, to add everything (if there are no unwanted files staged):  
 git add .  
  
2. Commit the change:  
 git commit -m "Add .gitignore to exclude log files and folder"  
 # This saves `.gitignore` into version history.

## Step 6: Final Verification

Run Git status again to confirm your working directory is clean.  
  
git status  
# Output: 'nothing to commit, working tree clean'.  
# This means `.gitignore` is in place and ignored files are not being tracked.