SIC-IOT702 / task 04 : git 01

Instractor : eng/ Mohamed Ahmed

Student name: Ahmed Mohamed Elsayed

Open-Ended Questions

What is the difference between git diff and git diff --staged?

git diff shows you the changes in your **working directory** that haven't been added to the staging area yet. For example, if you modify a file but don't run

git add, git diff will display those changes.

git diff --staged (also known as git diff --cached) shows you the changes that you have added to the **staging area** but have not yet committed. For example, after you run

git add main.py, git diff --staged will show the changes in that file, while git diff will show nothing.

If you accidentally staged a file, how would you remove it from the staging area but keep your modifications in the working directory?

You would use the command

git restore --staged <file>.

This is useful if you add a file to the staging area before you are finished with your changes. It allows you to unstage the file to continue working on it before committing, without losing any of your work.

Can you directly alias git commit as git ci using Git configuration?

Yes, you can directly create an alias for

git commit as git ci using Git's configuration. You can do this with the command

git config --global alias.ci commit, or by editing your global .gitconfig file directly.

What does the init.defaultBranch setting control in Git?

The init.defaultBranch setting controls the name of the initial branch created when you run git init. Teams might choose to set this to main instead of the traditional master to use more inclusive language.

If you have staged changes in main and then switch to a feature branch, what happens to those staged changes?

If you have staged changes on the

main branch and then switch to a feature branch, those staged changes will **move with you to the new branch**.

Git does this to prevent you from losing work that you haven't committed yet. It allows you to seamlessly move uncommitted changes between branches.

What is the difference between git switch -c <branch> and git checkout -b <branch>?

Both git switch -c <branch> and git checkout -b <branch> create a new branch and switch to it. The main difference is that git switch was introduced to create a safer and clearer command for branch switching.

git checkout is a multi-purpose command that can be used for many actions, like restoring files or switching to specific commits, which can be confusing. Separating these functions makes each command easier to understand and reduces the chance of making mistakes.

MCQ questions

Multiple Choice Questions

1. Which of the following is not a benefit of using a version control system?
    - **c) Automatically fixing bugs**
    - **Explanation:** Version control systems are excellent for tracking changes, collaborating with others, and reverting to previous versions of code. However, they cannot automatically correct logical errors or bugs within the code itself.

2. In a Centralized Version Control System (CVCS), where is the version database stored?
    - **b) On a single, central server**
    - **Explanation:** A centralized system relies on a single server to store all the project's history. Developers must connect to this server to commit or get updates.
3. Who developed Git and in what year?
    - **b) Linus Torvalds, 2005**
    - **Explanation:** Linus Torvalds, the creator of the Linux operating system, developed Git to manage the large-scale, distributed development of the Linux kernel.
4. Which command checks the installed Git version?
    - **c) git --version**
    - **Explanation:** This is the standard command-line convention for checking the version of many software tools, including Git.
5. Which term refers to copying an existing remote repository to your local machine?
    - **c) Clone**
    - **Explanation:** When you "clone" a repository, you download a complete copy of the project, including all its files and the full commit history.
6. Which area in Git acts as an intermediate space between the working directory and the repository?
    - **b) The staging area (index)**
    - **Explanation:** The staging area is where you prepare changes to be included in your next commit. You add files to this area with git add before committing them.
7. Which command renames the current branch to main?
    - **b) git branch -M main**
    - **Explanation:** The git branch command is used to list, create, or delete branches. The -M flag specifically renames a branch, which is often used to change the default branch name.

8. Which command is used to download an existing remote repository for the first time?
    - **a) git clone**
    - **Explanation:** This command is used once at the beginning of a project to create a local copy of a remote repository. After that, you'll use other commands like git pull to get new updates.
9. What is the purpose of a pull request in GitHub?
    - **b) To propose merging changes from one branch into another**
    - **Explanation:** A pull request is a formal request for a project maintainer to review and integrate your changes into the main codebase.
10. If Peter wants to push changes to Daniel's repository but doesn't have permission, what should Daniel do?
    - **b) Add Peter as a collaborator**
    - **Explanation:** Adding Peter as a collaborator is the simplest way to grant him the necessary permissions to push his changes directly to the repository.

Practice project

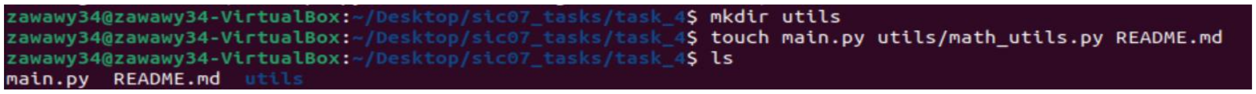
Scenario

You are creating a small Python project with these files:

• main.py

• utils/math_utils.py

• README.md



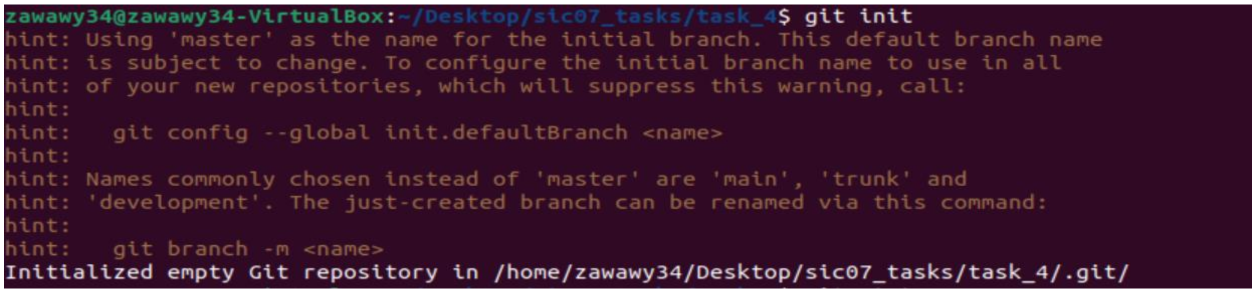
```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ mkdir utils
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ touch main.py utils/math_utils.py README.md
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ ls
main.py  README.md  utils
```

Tasks

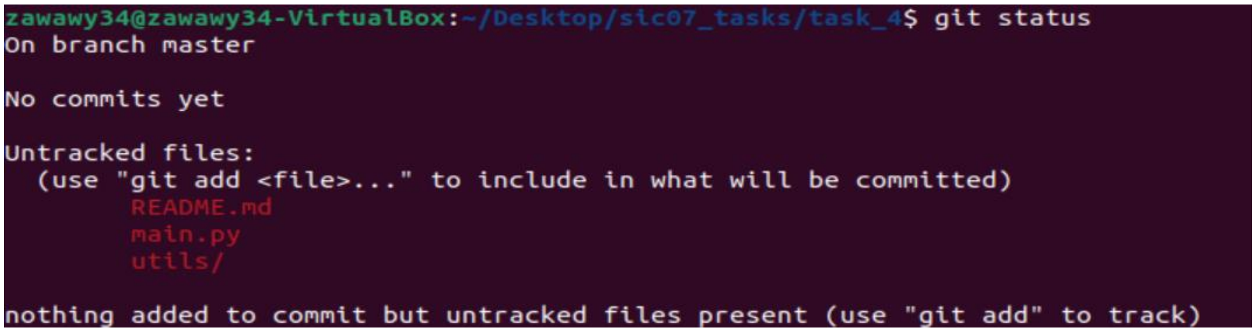
1. Setup

• Initialize a new Git repo.

• Configure your **default editor** (pick nano, vim, or code --wait).

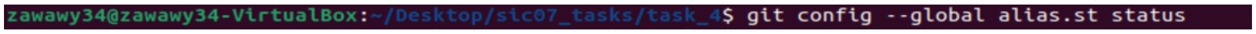


```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/zawawy34/Desktop/sic07_tasks/task_4/.git/
```

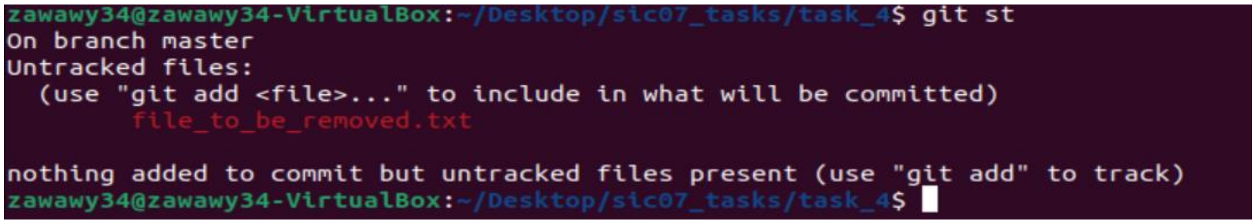


```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        main.py
        utils/

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

• Add an **alias** so you can type st instead of the full status command.



```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git config --global alias.st status
```

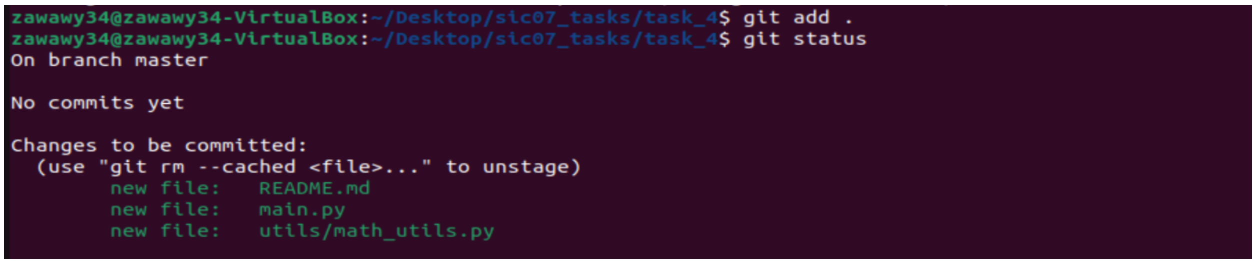


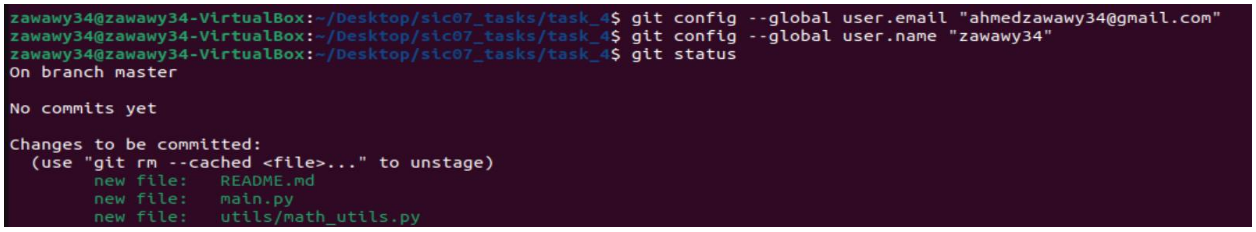
```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git st
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file_to_be_removed.txt

nothing added to commit but untracked files present (use "git add" to track)
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$
```

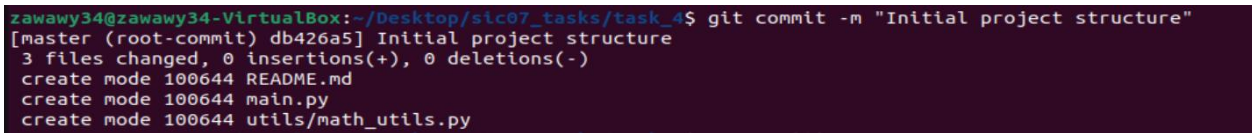
2. First Commit

• Add all project files and make your **first commit**: *"Initial project structure"*.

• Explore the .git/objects/ directory. (Hint: use a Git plumbing command to read the content of a blob or tree (cat-file)).


```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git add .
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   main.py
        new file:   utils/math_utils.py
```

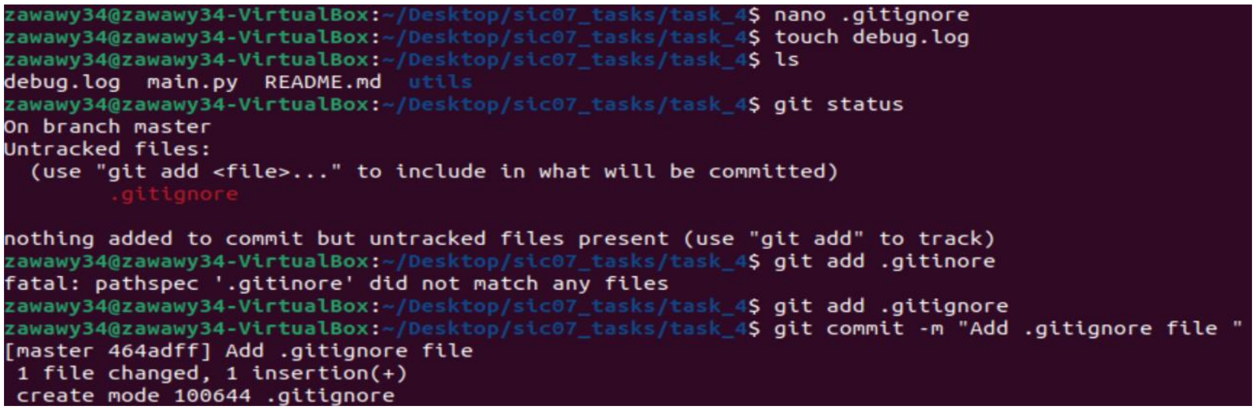

```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git config --global user.email "ahmedzawawy34@gmail.com"
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git config --global user.name "zawawy34"
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   main.py
        new file:   utils/math_utils.py
```


```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git commit -m "Initial project structure"
[master (root-commit) db426a5] Initial project structure
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
 create mode 100644 main.py
 create mode 100644 utils/math_utils.py
```

3. Ignore Files

• Create a rule to ignore all log files.

• Test by creating a debug.log file and check that Git ignores it.


```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ nano .gitignore
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ touch debug.log
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ ls
debug.log  main.py  README.md  utils
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git status
On branch master
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)
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git add .gitinore
fatal: pathspec '.gitinore' did not match any files
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git add .gitignore
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git commit -m "Add .gitignore file "
[master 464adff] Add .gitignore file
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore
```

4. New Feature (Branching)

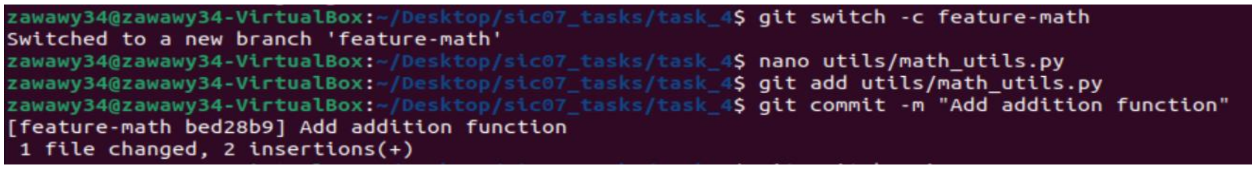
• Create a new branch called feature-math.

• Inside utils/math_utils.py, add a function:

def add(a, b):

return a + b

• Commit this change to the branch.



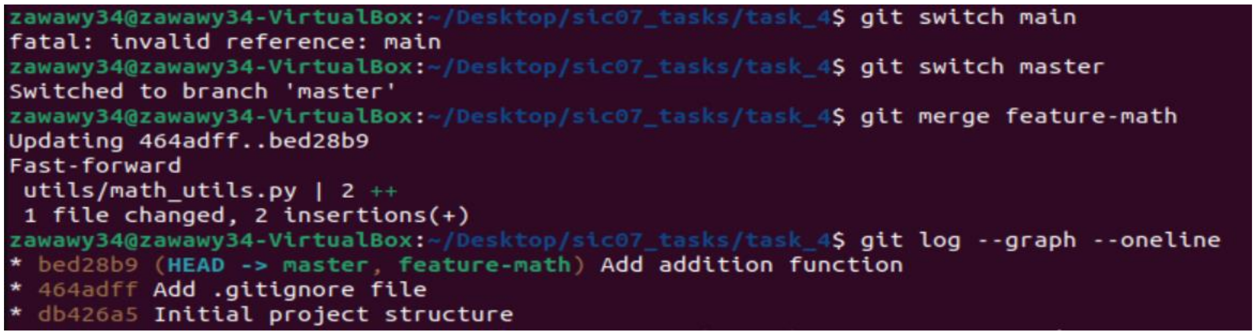
```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git switch -c feature-math
Switched to a new branch 'feature-math'
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ nano utils/math_utils.py
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git add utils/math_utils.py
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git commit -m "Add addition function"
[feature-math bed28b9] Add addition function
 1 file changed, 2 insertions(+)
```

5. Merging

• Switch back to main.

• Merge the branch into main.

• Check if the merge was fast-forward or a 3-way merge and what is the difference between the two ways and show your answer using a diagram or using (log command <-- bonus) ?



```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git switch main
fatal: invalid reference: main
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git switch master
Switched to branch 'master'
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git merge feature-math
Updating 464adff..bed28b9
Fast-forward
 utils/math_utils.py | 2 ++
 1 file changed, 2 insertions(+)
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git log --graph --oneline
* bed28b9 (HEAD -> master, feature-math) Add addition function
* 464adff Add .gitignore file
* db426a5 Initial project structure
```

6. Undo / Unstage

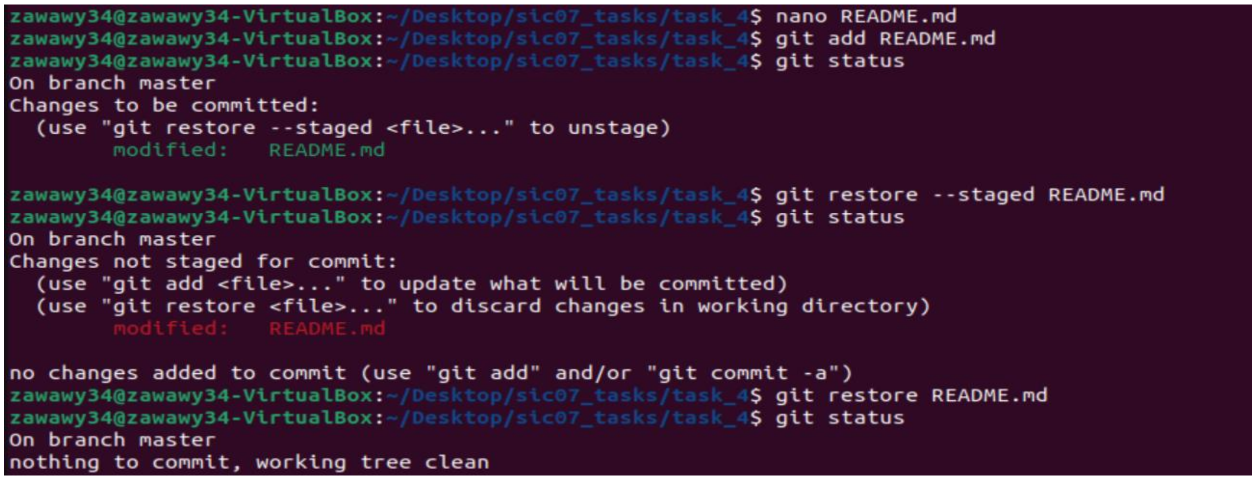
• Edit README.md (e.g., add "This is a math project") and stage it.

• Oops! Unstage it without deleting your changes.

• Then discard your changes completely.

• Explain for me what is the difference between restore –staged, --worktree and rm –cached ?

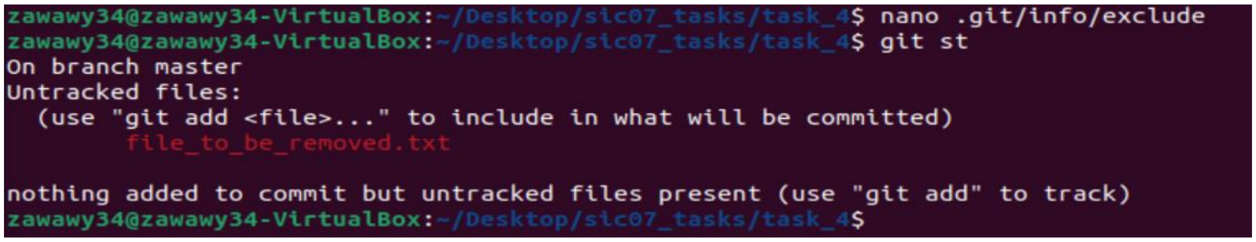
And show your explanation in your terminal <3.



```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ nano README.md
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git add README.md
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git restore --staged README.md
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git restore README.md
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git status
On branch master
nothing to commit, working tree clean
```

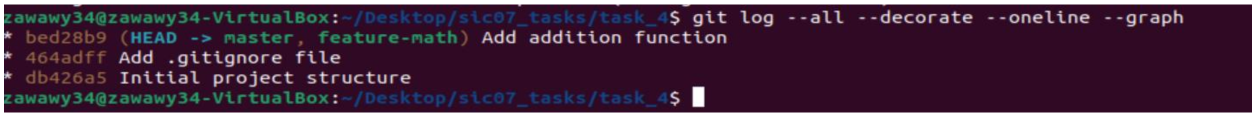
7. Bonus Challenge

• Ignore a file using .git/info/exclude instead of .gitignore.



```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ nano .git/info/exclude
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git st
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file_to_be_removed.txt

nothing added to commit but untracked files present (use "git add" to track)
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$
```

• Visualize the commit history as a **graph** (Hint: compact one-line graph view).



```
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$ git log --all --decorate --oneline --graph
* bed28b9 (HEAD -> master, feature-math) Add addition function
* 464adff Add .gitignore file
* db426a5 Initial project structure
zawawy34@zawawy34-VirtualBox:~/Desktop/sic07_tasks/task_4$
```