SEPM EXP NO: 3

TO PERFORM VARIOUS GIT OPERATIONS ON LOCAL AND REMOTE REPOSITORIES USING GIT CHEAT SHEET

Mustafa Sarangpurwala
T21-88
AI&DS

Theory:

Git is a distributed version control system that allows developers to track changes, collaborate, and manage source code efficiently. Git provides numerous commands to handle local and remote repositories.

1. Setting Up Git

Before performing Git operations, configure Git with your details:

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

git config --global user.email "your.email@example.com" Verify the configuration:

git config --list

2. Initializing a Git Repository To create a new Git repository: git init

This initializes a new repository in the current directory.

3. Cloning a Repository To clone a remote repository: git clone <repository_url> Example:

git clone https://github.com/your-username/repository.git

4. Staging and Committing Changes

- To check the status of the working directory:
- git status
- To add files to the staging area:

- git add <file_name> or to add all changes:

git add .

- To commit changes with a message:
- git commit -m "Your commit message" 5. Viewing Commit History To view commit logs:

git log

For a compact version: git
log --oneline

6. Branching in Git

- To create a new branch:
- git branch <branch_name>
- To switch to another branch: □ git checkout <branch_name>
- To create and switch to a new branch simultaneously:
- git checkout -b <branch_name> □ To view all branches:
- git branch

7. Merging Branches

- First, switch to the main branch:
- git checkout main
- Merge a branch into the main branch:
- git merge <branch_name> 8. Pushing Changes to Remote Repository □ To push changes to GitHub: □ git push origin <branch_name> □ If pushing for the first time:
- git push --set-upstream origin <branch_name>

9. Pulling Changes from Remote Repository To fetch
and merge changes from a remote repository:

git pull origin <branch_name>

10. Handling Merge Conflicts If a
merge conflict occurs:

1. Open conflicting files and resolve issues manually.
2. Add resolved files to the staging area:

3. git add <file_name>
4. Commit the resolved changes:
5. git commit -m "Resolved merge conflict"

11. Undoing Changes

- To undo changes before staging:
- git checkout -- <file_name> □ To unstage a file:
- git reset HEAD <file_name>
- To revert the last commit:
- git revert HEAD

12. Deleting a Branch

- To delete a local branch: □ git branch -d <branch_name> □ To delete a remote branch:
- git push origin --delete <branch_name>

13. Creating and Using a .gitignore File

A .gitignore file is used to ignore specific files or directories:

```
echo "node_modules/" >> .gitignore
git add .gitignore git commit -m
"Added .gitignore file"
```

14. Checking Differences in Files

- To compare working directory changes:
- git diff
- To compare staged changes:
- git diff --staged

15. Stashing Changes

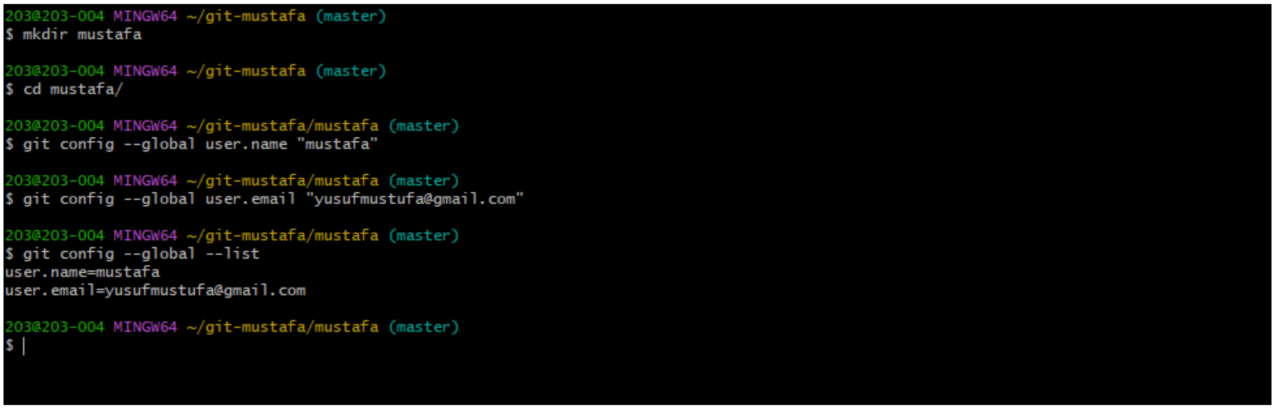
To temporarily save uncommitted changes:

```
git stash
```

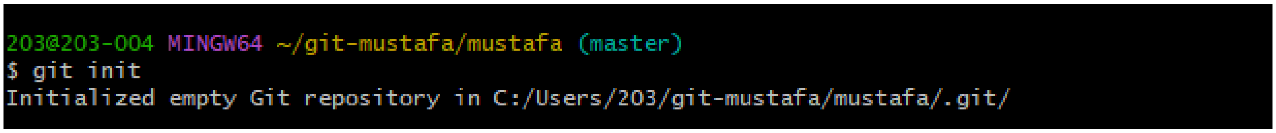
To apply the stashed changes:

```
git stash apply
```

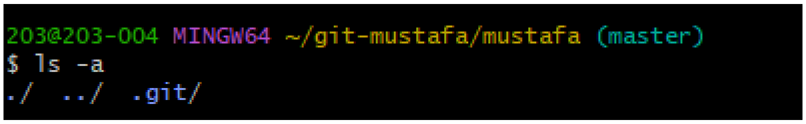
Output:



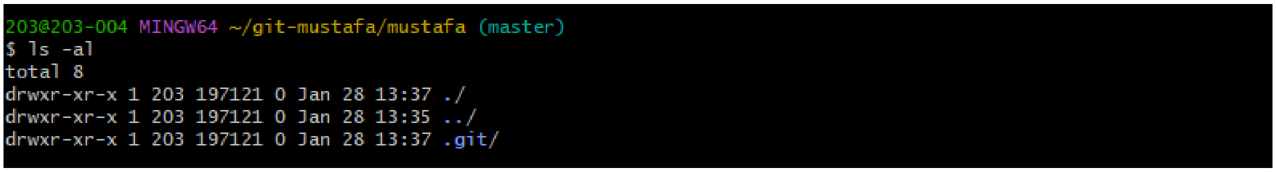
```
203@203-004 MINGW64 ~/git-mustafa (master)
$ mkdir mustafa

203@203-004 MINGW64 ~/git-mustafa (master)
$ cd mustafa/

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git config --global user.name "mustafa"

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git config --global user.email "yusufmustufa@gmail.com"

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git config --global --list
user.name=mustafa
user.email=yusufmustufa@gmail.com

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ |
```



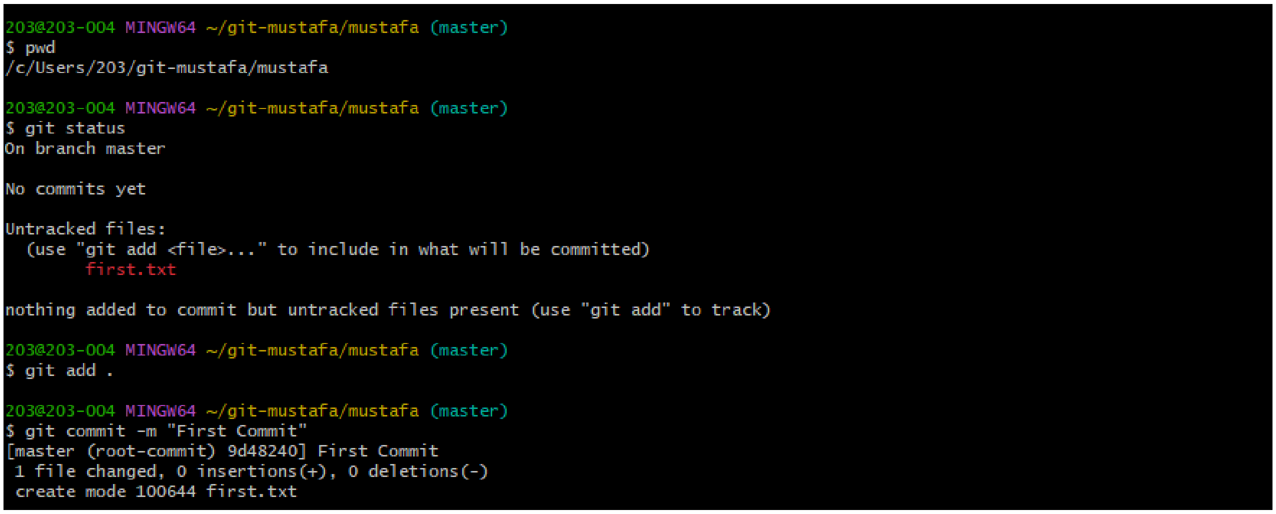
```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git init
Initialized empty Git repository in C:/Users/203/git-mustafa/mustafa/.git/
```



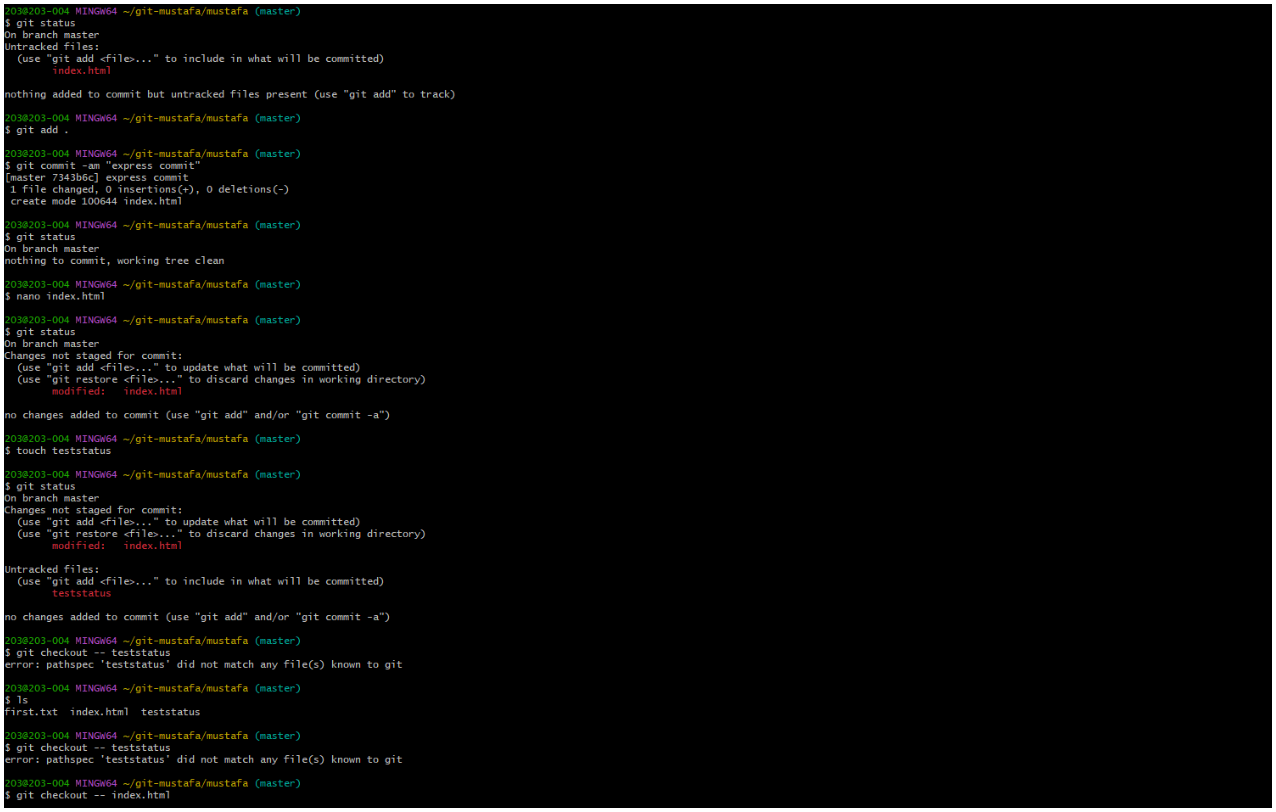
```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ ls -a
./  ../  .git/
```



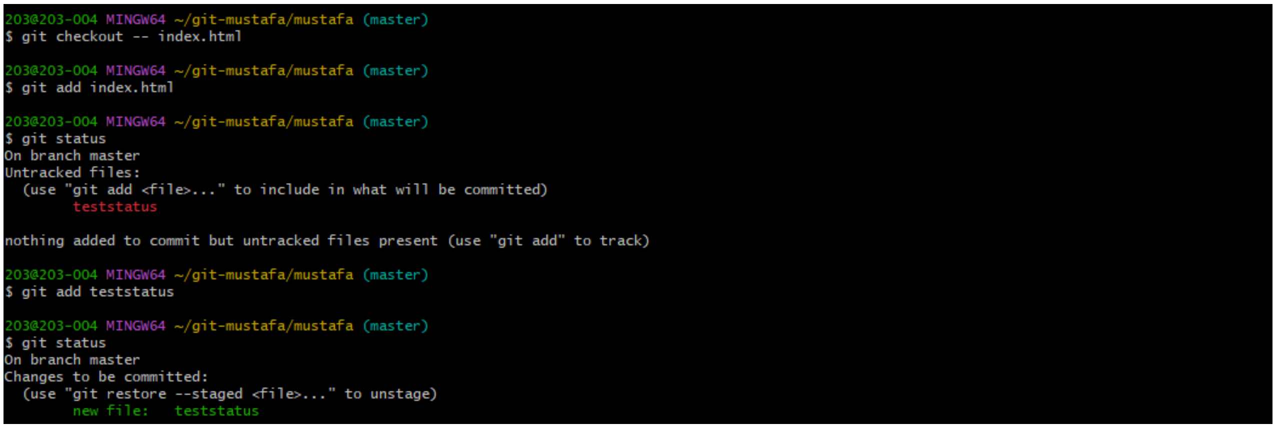
```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ ls -al
total 8
drwxr-xr-x 1 203 197121 0 Jan 28 13:37 ./
drwxr-xr-x 1 203 197121 0 Jan 28 13:35 ../
drwxr-xr-x 1 203 197121 0 Jan 28 13:37 .git/
```



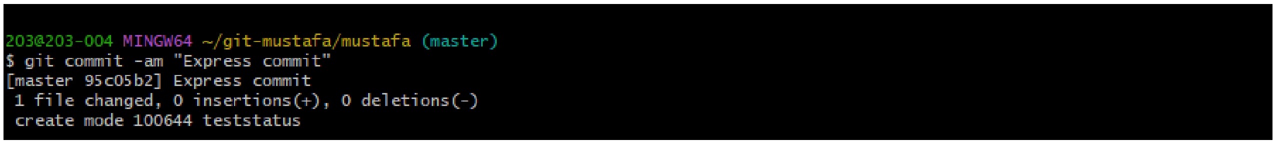
```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ pwd
/c/Users/203/git-mustafa/mustafa

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        first.txt

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

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git add .

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git commit -m "First Commit"
[master (root-commit) 9d48240] First Commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 first.txt
```



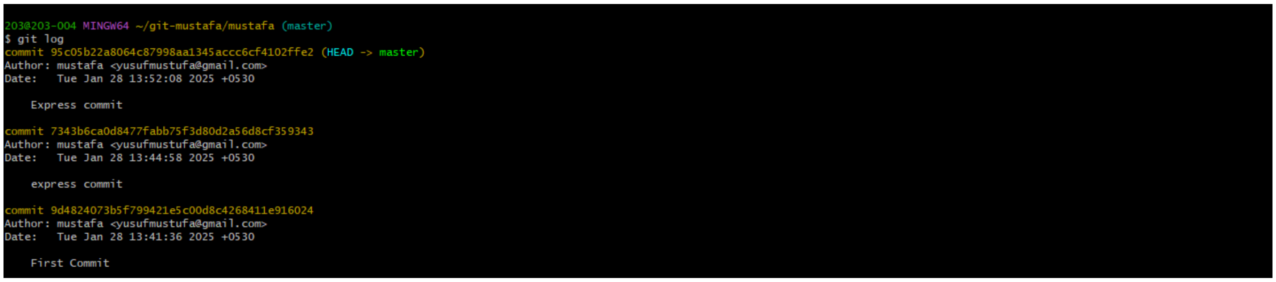
```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

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

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git add .

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git commit -am "express commit"
[master 7343b6c] express commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git status
On branch master
nothing to commit, working tree clean

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ nano index.html

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ 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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ touch teststatus

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ 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:   index.html

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

no changes added to commit (use "git add" and/or "git commit -a")

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git checkout -- teststatus
error: pathspec 'teststatus' did not match any file(s) known to git

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ ls
first.txt  index.html  teststatus

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git checkout -- teststatus
error: pathspec 'teststatus' did not match any file(s) known to git

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git checkout -- index.html
```



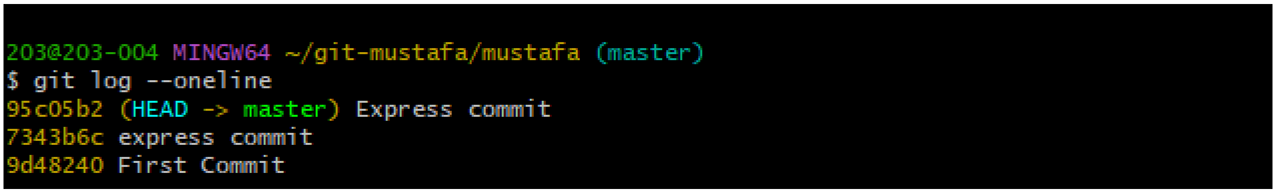
```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git checkout -- index.html

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git add index.html

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        teststatus

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

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git add teststatus

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   teststatus
```



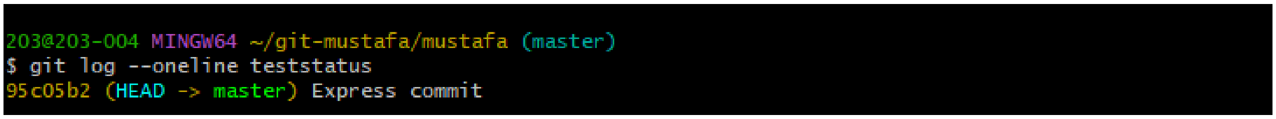
```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git commit -am "Express commit"
[master 95c05b2] Express commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 teststatus
```



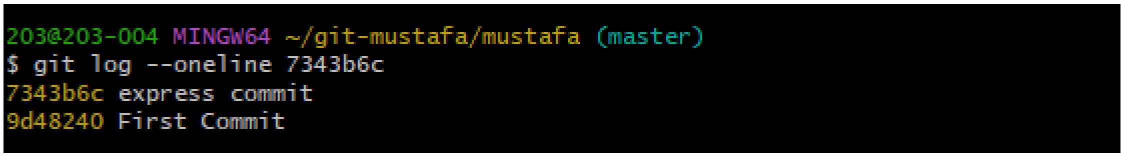
```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git log
commit 95c05b22a8064c87998aa1345accc6cf4102ffe2 (HEAD -> master)
Author: mustafa <yusufmustufa@gmail.com>
Date:   Tue Jan 28 13:52:08 2025 +0530

    Express commit

commit 7343b6ca0d8477fabb75f3d80d2a56d8cf359343
Author: mustafa <yusufmustufa@gmail.com>
Date:   Tue Jan 28 13:44:58 2025 +0530

    express commit

commit 9d4824073b5f799421e5c00d8c4268411e916024
Author: mustafa <yusufmustufa@gmail.com>
Date:   Tue Jan 28 13:41:36 2025 +0530

    First Commit
```



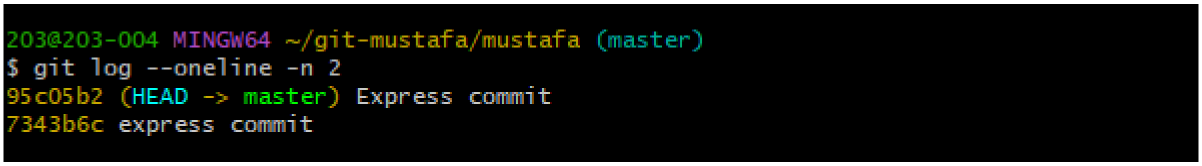
```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git log --oneline
95c05b2 (HEAD -> master) Express commit
7343b6c express commit
9d48240 First Commit
```



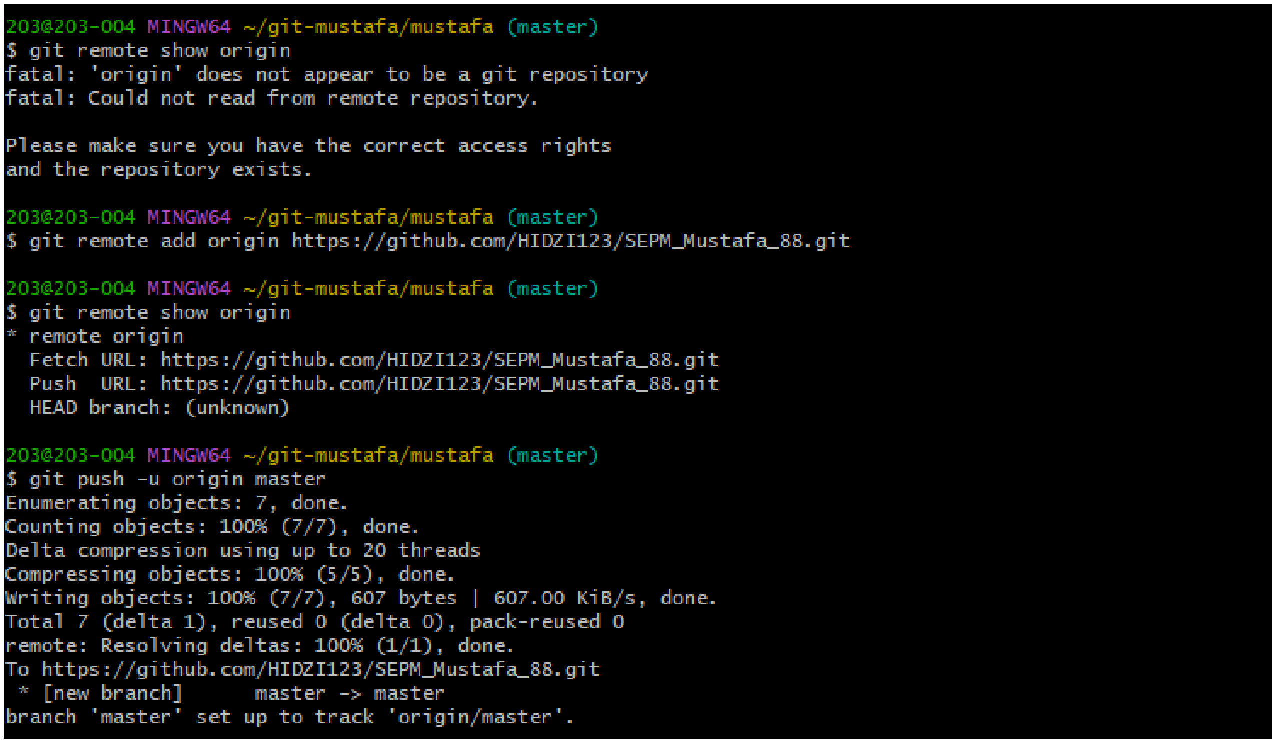
```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git log --oneline teststatus
95c05b2 (HEAD -> master) Express commit
```



```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git log --oneline 7343b6c
7343b6c express commit
9d48240 First Commit
```



```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git log --oneline -n 2
95c05b2 (HEAD -> master) Express commit
7343b6c express commit
```



```
203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git remote show origin
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git remote add origin https://github.com/HIDZI123/SEPM_Mustafa_88.git

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/HIDZI123/SEPM_Mustafa_88.git
  Push  URL: https://github.com/HIDZI123/SEPM_Mustafa_88.git
  HEAD branch: (unknown)

203@203-004 MINGW64 ~/git-mustafa/mustafa (master)
$ git push -u origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 607 bytes | 607.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/HIDZI123/SEPM_Mustafa_88.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

Conclusion

This experiment demonstrated various Git operations, including repository initialization, branching, merging, pushing, pulling, and resolving conflicts. These commands help in efficient version control and collaboration in software development projects.