Name: Maira Malik

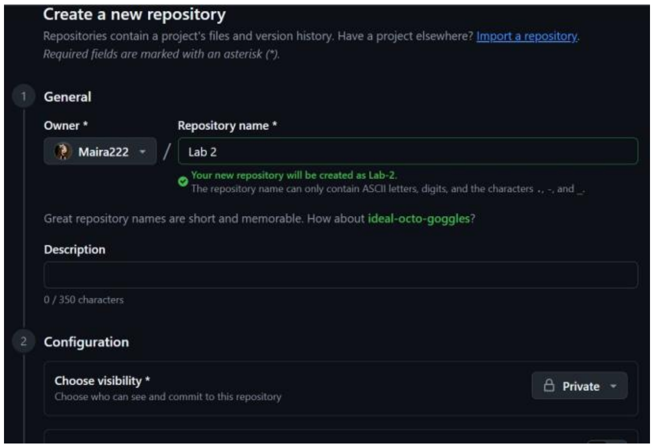
Reg.no: 2023-BSE-040

Subject: Cloud Computing

LAB # 02

Task 1: Create Private GitHub Repository

Create a new private repository named Lab2 on GitHub.



Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? Import a repository.
Required fields are marked with an asterisk (*).

1 General

Owner * Maira222 / Repository name * Lab 2

Your new repository will be created as Lab-2.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

Great repository names are short and memorable. How about ideal-octo-goggles?

Description

0 / 350 characters

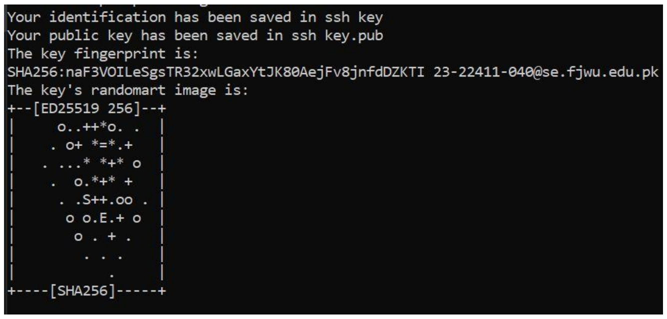
2 Configuration

Choose visibility *
Choose who can see and commit to this repository — Private

Task 2: Connect Repository via SSH

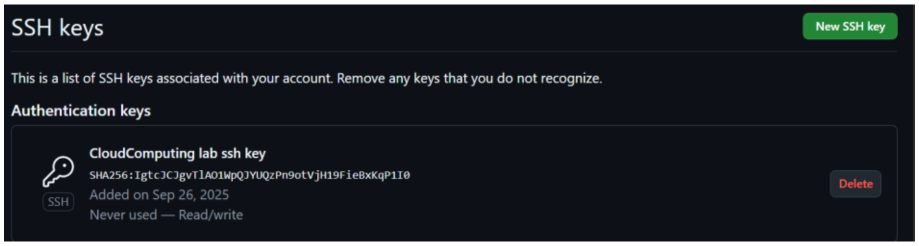
1. Generate a new SSH key using PowerShell:

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```



```
Your identification has been saved in ssh key
Your public key has been saved in ssh key.pub
The key fingerprint is:
SHA256:naF3VOILeSgsTR32xwLGaxYtJK80AejFv8jnfdDZKTI 23-22411-040@se.fjwu.edu.pk
The key's randomart image is:
+--[ED25519 256]--+
|     o..++*o. .  |
|    . o+ *=*.+   |
|   . ...* *+* o  |
|    .  o.*+* +   |
|     . .S++.oo . |
|      o o.E.+ o  |
|       o . + .   |
|         . . .   |
|           .     |
+----[SHA256]-----+
```

2. Add your SSH public key to GitHub (Settings > SSH and GPG keys).



SSH keys

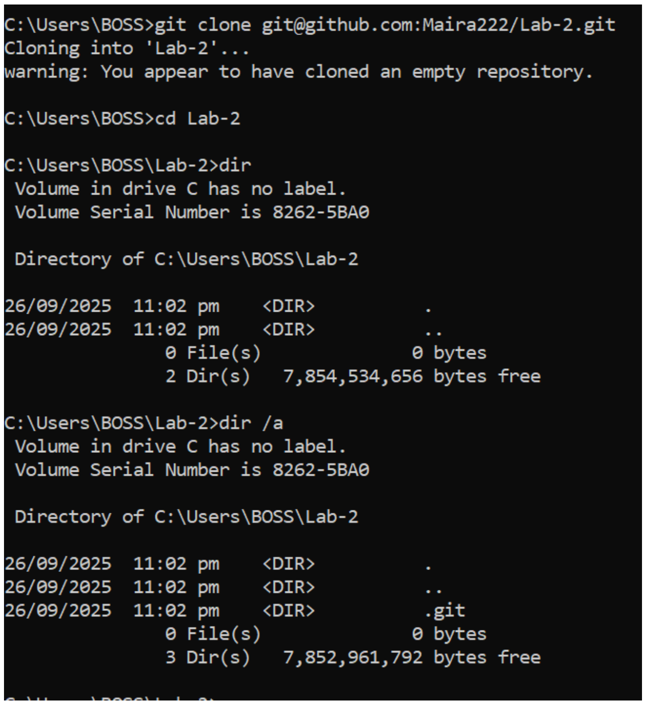
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication keys

CloudComputing lab ssh key
SHA256:IgtcJCJgvTlAO1WpQJYUQzPn9otVjH19FieBxKqP1I0
Added on Sep 26, 2025
Never used — Read/write

3. Clone your `Lab2` repo using SSH.

```
git clone git@github.com:<yourusername>/Lab2.git
```

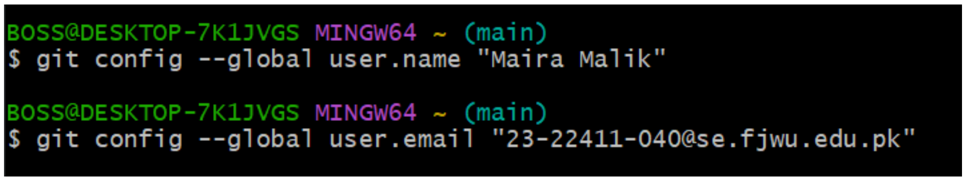


```
C:\Users\BOSS>git clone git@github.com:Maira222/Lab-2.git
Cloning into 'Lab-2'...
warning: You appear to have cloned an empty repository.

C:\Users\BOSS>cd Lab-2

C:\Users\BOSS\Lab-2>dir
 Volume in drive C has no label.
 Volume Serial Number is 8262-5BA0

 Directory of C:\Users\BOSS\Lab-2

26/09/2025  11:02 pm    <DIR>          .
26/09/2025  11:02 pm    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)   7,854,534,656 bytes free

C:\Users\BOSS\Lab-2>dir /a
 Volume in drive C has no label.
 Volume Serial Number is 8262-5BA0

 Directory of C:\Users\BOSS\Lab-2

26/09/2025  11:02 pm    <DIR>          .
26/09/2025  11:02 pm    <DIR>          ..
26/09/2025  11:02 pm    <DIR>          .git
               0 File(s)              0 bytes
               3 Dir(s)   7,852,961,792 bytes free
```

Task 3: Configure Git Username and Email

1. Set up your Git identity (this ensures all commits are linked to you):

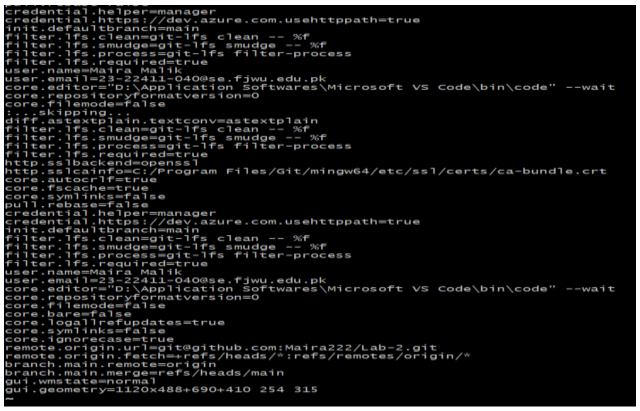
```
git config --global user.name "Your Name"

git config --global user.email "your_email@example.com"
```



```
BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ git config --global user.name "Maira Malik"

BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ git config --global user.email "23-22411-040@se.fjwu.edu.pk"
```

2. Verify your configuration:

```
git config --list
```



```
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=main
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
user.name=Maira Malik
user.email=23-22411-040@se.fjwu.edu.pk
core.editor="D:\Application Softwares\Microsoft VS Code\bin\code" --wait
core.repositoryformatversion=0
core.filemode=false
:...skipping...
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=main
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
user.name=Maira Malik
user.email=23-22411-040@se.fjwu.edu.pk
core.editor="D:\Application Softwares\Microsoft VS Code\bin\code" --wait
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
remote.origin.url=git@github.com:Maira222/Lab-2.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.main.remote=origin
branch.main.merge=refs/heads/main
gui.wmstate=normal
gui.geometry=1120x488+690+410 254 315
~
```

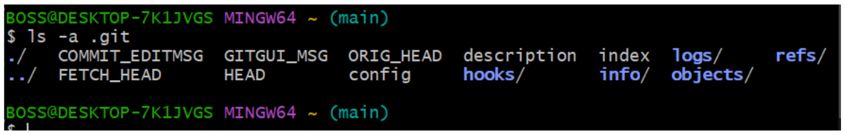
Task 4: Explore the .git Folder

Navigate into your cloned repository folder.

Show hidden files and locate the `.git` directory.

Explore what's inside using:

```
ls -a .git
```

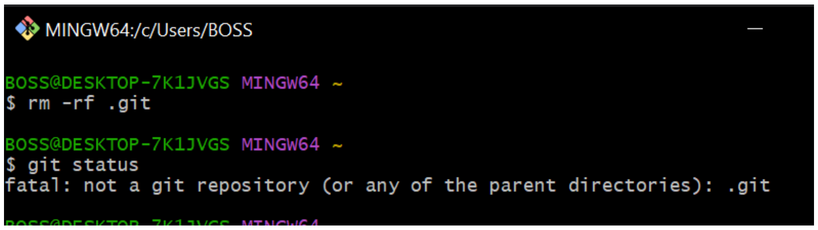


```
BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ ls -a .git
./   COMMIT_EDITMSG  GITGUI_MSG  ORIG_HEAD  description  index  logs/     refs/
../  FETCH_HEAD      HEAD        config     hooks/       info/  objects/

BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
```

Task 5: Local Repository Management

1. Delete the existing `.git` folder from your cloned repo using Git Bash:

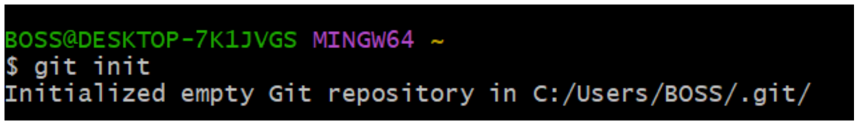
```
rm -rf .git
```



```
MINGW64:/c/Users/BOSS

BOSS@DESKTOP-7K1JVGS MINGW64 ~
$ rm -rf .git

BOSS@DESKTOP-7K1JVGS MINGW64 ~
$ git status
fatal: not a git repository (or any of the parent directories): .git
```

2. Re-initialize the local git repository:

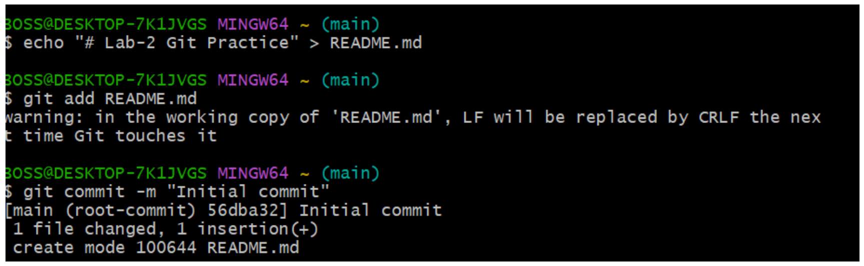
```
git init
```



```
BOSS@DESKTOP-7K1JVGS MINGW64 ~
$ git init
Initialized empty Git repository in C:/Users/BOSS/.git/
```

3. Add a file named `README.md` and commit it:

```
echo "# Lab2 Git Practice" > README.md

git add README.md

git commit -m "Initial commit"
```

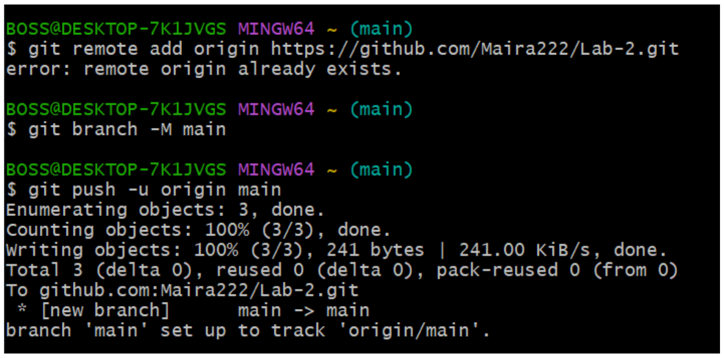


```
BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ echo "# Lab-2 Git Practice" > README.md

BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the nex
t time Git touches it

BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ git commit -m "Initial commit"
[main (root-commit) 56dba32] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

4. Connect your local repo to GitHub and push:

```
git remote add origin git@github.com:<yourusername>/Lab2.git
```

```
git push -u origin main
```



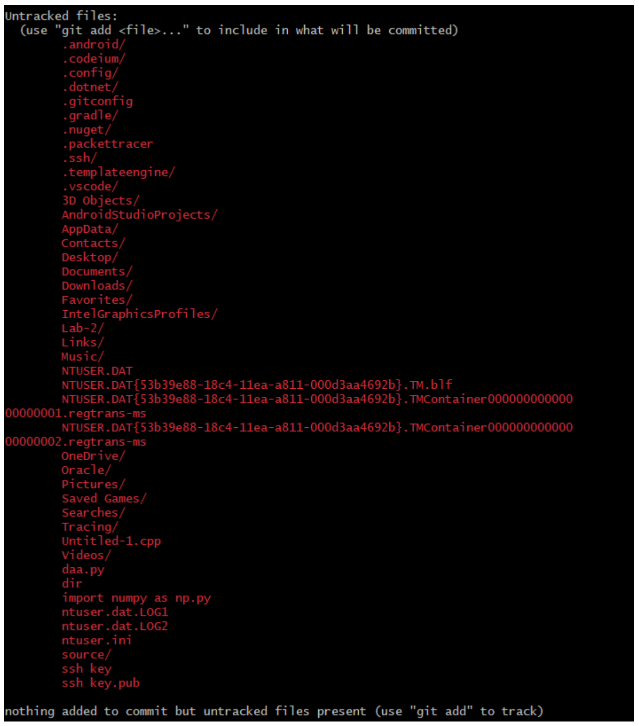
```
BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ git remote add origin https://github.com/Maira222/Lab-2.git
error: remote origin already exists.

BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ git branch -M main

BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 241 bytes | 241.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:Maira222/Lab-2.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Task 6: File Status & Staging

1. Create a new file `notes.txt` and write a note.

Check status:

```
git status
```

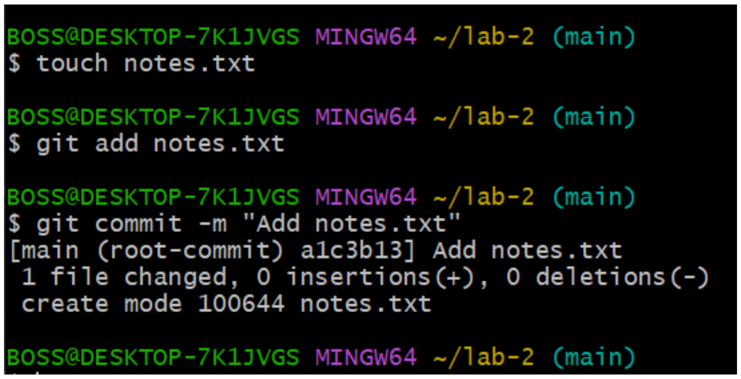


```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .android/
        .codeium/
        .config/
        .dotnet/
        .gitconfig
        .gradle/
        .nuget/
        .packettracer
        .ssh/
        .templateengine/
        .vscode/
        3D Objects/
        AndroidStudioProjects/
        AppData/
        Contacts/
        Desktop/
        Documents/
        Downloads/
        Favorites/
        IntelGraphicsProfiles/
        Lab-2/
        Links/
        Music/
        NTUSER.DAT
        NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TM.blf
        NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TMContainer000000000000
00000001.regtrans-ms
        NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TMContainer000000000000
00000002.regtrans-ms
        OneDrive/
        Oracle/
        Pictures/
        Saved Games/
        Searches/
        Tracing/
        Untitled-1.cpp
        Videos/
        daa.py
        dir
        import numpy as np.py
        ntuser.dat.LOG1
        ntuser.dat.LOG2
        ntuser.ini
        source/
        ssh key
        ssh key.pub

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

Stage and commit:

```
git add notes.txt
```

```
git commit -m "Add notes.txt"
```



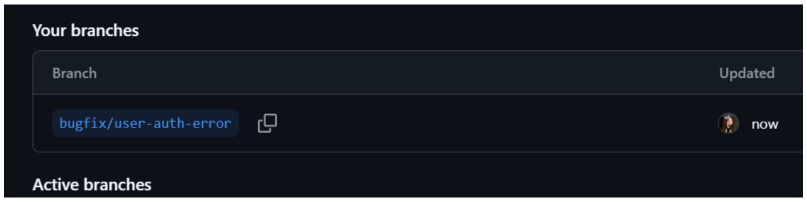
```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ touch notes.txt

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git add notes.txt

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git commit -m "Add notes.txt"
[main (root-commit) a1c3b13] Add notes.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 notes.txt

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
```

Task 7: Branch Creation Using GitHub GUI

On GitHub (web interface), create a branch named `bugfix/user-auth-error`.

Pull the branch to your local repository to sync.

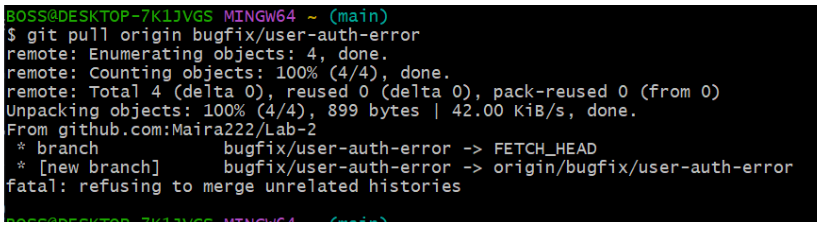
```
git pull origin bugfix/user-auth-error
```



Your branches

Branch	Updated
bugfix/user-auth-error	now

Active branches



```
BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ git pull origin bugfix/user-auth-error
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (4/4), 899 bytes | 42.00 KiB/s, done.
From github.com:Maira222/Lab-2
 * branch            bugfix/user-auth-error -> FETCH_HEAD
 * [new branch]      bugfix/user-auth-error -> origin/bugfix/user-auth-error
fatal: refusing to merge unrelated histories
```

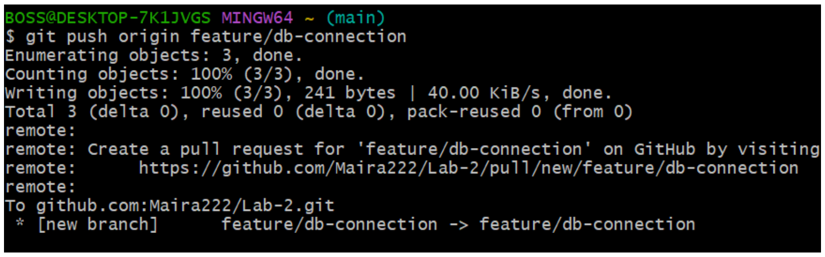
Task 8: Branch Creation and Push Using Git Bash

1. Create a branch named `feature/db-connection` using Git Bash:

```
git checkout -b feature/db-connection
```

2. Push the branch to the remote repository:

```
git push origin feature/db-connection
```

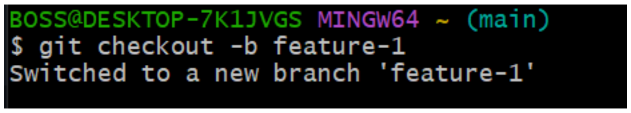


```
BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ git push origin feature/db-connection
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 241 bytes | 40.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature/db-connection' on GitHub by visiting
remote:      https://github.com/Maira222/Lab-2/pull/new/feature/db-connection
remote:
To github.com:Maira222/Lab-2.git
 * [new branch]      feature/db-connection -> feature/db-connection
```

Task 9: Branching & Merging

1. Create and switch to a branch `feature-1`:

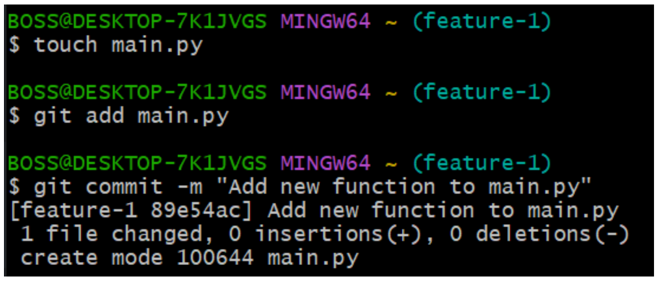
```
git checkout -b feature-1
```



```
BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ git checkout -b feature-1
Switched to a new branch 'feature-1'
```

2. Modify `main.py` (add a function) and commit.

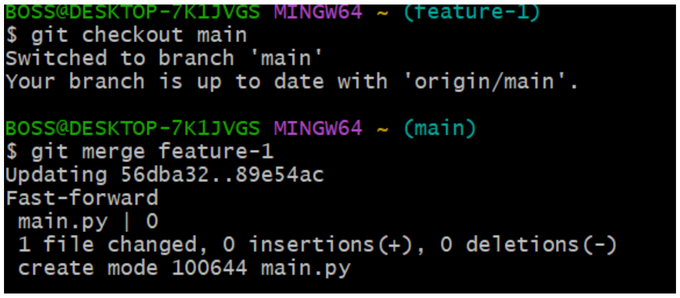
```
git add main.py

git commit -m "Add new function to main.py"
```



```
BOSS@DESKTOP-7K1JVGS MINGW64 ~ (feature-1)
$ touch main.py

BOSS@DESKTOP-7K1JVGS MINGW64 ~ (feature-1)
$ git add main.py

BOSS@DESKTOP-7K1JVGS MINGW64 ~ (feature-1)
$ git commit -m "Add new function to main.py"
[feature-1 89e54ac] Add new function to main.py
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 main.py
```

3. Switch back to `main` and merge:

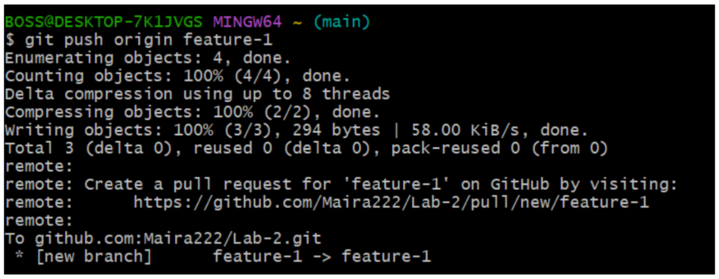
```
git checkout main

git merge feature-1
```



```
BOSS@DESKTOP-7K1JVGS MINGW64 ~ (feature-1)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ git merge feature-1
Updating 56dba32..89e54ac
Fast-forward
 main.py | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 main.py
```

4. Push all branches:

```
git push origin main

git push origin feature-1
```



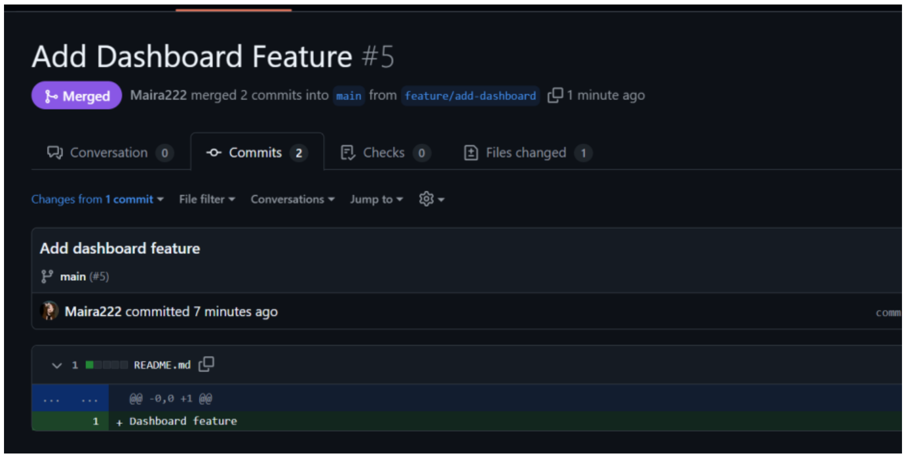
```
BOSS@DESKTOP-7K1JVGS MINGW64 ~ (main)
$ git push origin feature-1
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 294 bytes | 58.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-1' on GitHub by visiting:
remote:      https://github.com/Maira222/Lab-2/pull/new/feature-1
remote:
To github.com:Maira222/Lab-2.git
 * [new branch]      feature-1 -> feature-1
```

Task 10: Pull Request and Branch Review (GitHub GUI)

On GitHub, create a Pull Request from the branch `feature/db-connection` to `main`.

Review the Pull Request and merge it using the GitHub GUI.

After merging, delete the `feature/db-connection` branch using the GitHub GUI.


Add Dashboard Feature #5

Merged Maira222 merged 2 commits into `main` from `feature/add-dashboard` 1 minute ago

Conversation 0 | Commits 2 | Checks 0 | Files changed 1

Changes from 1 commit ▾ File filter ▾ Conversations ▾ Jump to ▾

Add dashboard feature
main (#5)
Maira222 committed 7 minutes ago

```
1 README.md
@@ -0,0 +1 @@
+ Dashboard feature
```

Task 11: Detailed Branch Strategy (Develop/Staging)

Create the following branches to simulate a professional branching strategy:

`develop`

`staging`

`feature/*`

`bugfix/*`

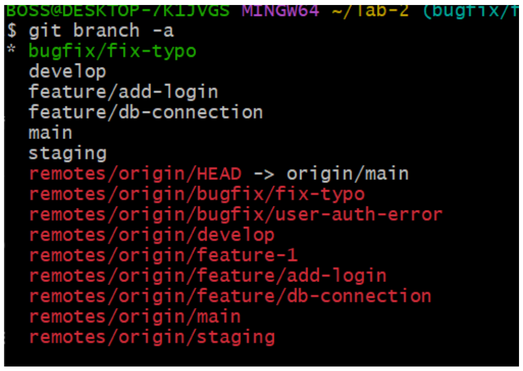
Example workflow:

Developers work on `feature/*` and `bugfix/*` branches.

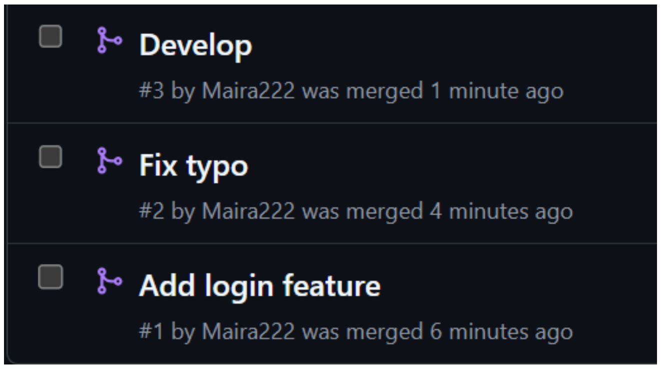
Merge into `develop` after completion.

`develop` is merged into `staging` for testing.

Finally, `staging` is merged into `main` (production).



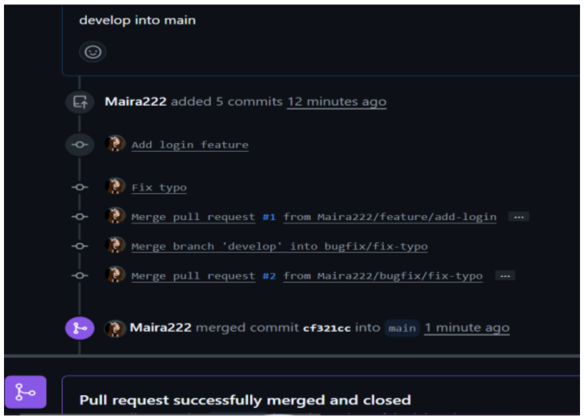
```
BOSS@DESKTOP-7KIJVGS MINGW64 ~/lab-2 (bugfix/f
$ git branch -a
* bugfix/fix-typo
  develop
  feature/add-login
  feature/db-connection
  main
  staging
  remotes/origin/HEAD -> origin/main
  remotes/origin/bugfix/fix-typo
  remotes/origin/bugfix/user-auth-error
  remotes/origin/develop
  remotes/origin/feature-1
  remotes/origin/feature/add-login
  remotes/origin/feature/db-connection
  remotes/origin/main
  remotes/origin/staging
```



Develop
#3 by Maira222 was merged 1 minute ago

Fix typo
#2 by Maira222 was merged 4 minutes ago

Add login feature
#1 by Maira222 was merged 6 minutes ago



develop into main

Maira222 added 5 commits 12 minutes ago

- Add login feature
- Fix typo
- Merge pull request #1 from Maira222/feature/add-login
- Merge branch 'develop' into bugfix/fix-typo
- Merge pull request #2 from Maira222/bugfix/fix-typo

Maira222 merged commit cf321cc into `main` 1 minute ago

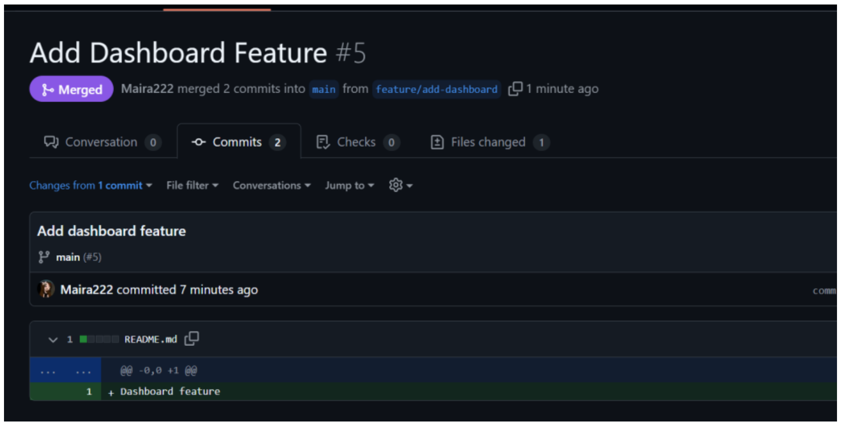
Pull request successfully merged and closed

Task 12: Code Review Workflow

Create a Pull Request (PR) / Merge Request (MR):

From a feature branch into `main`.

Add a clear title and description summarizing the changes.


Add Dashboard Feature #5

Merged Maira222 merged 2 commits into main from feature/add-dashboard 1 minute ago

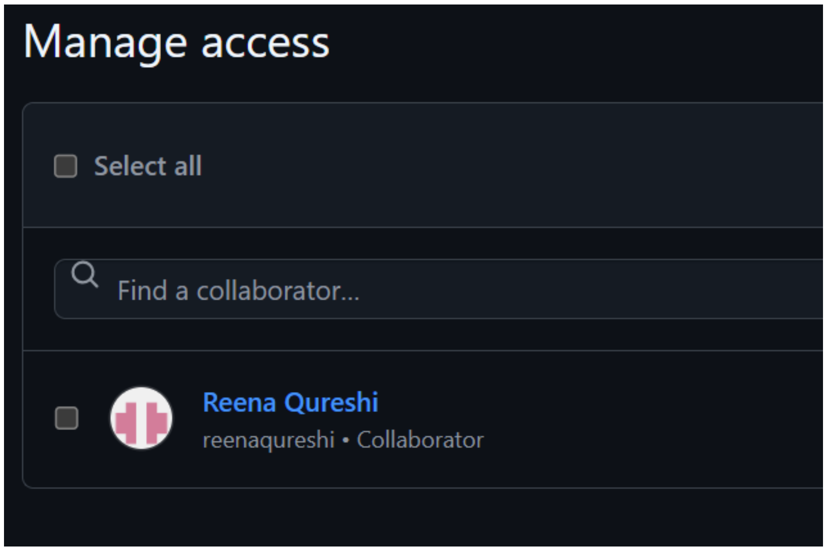
Add dashboard feature

main (#5)

Maira222 committed 7 minutes ago

```
1 README.md
@@ -0,0 +1 @@
+ Dashboard feature
```

Assign a reviewer (teammate or second account).

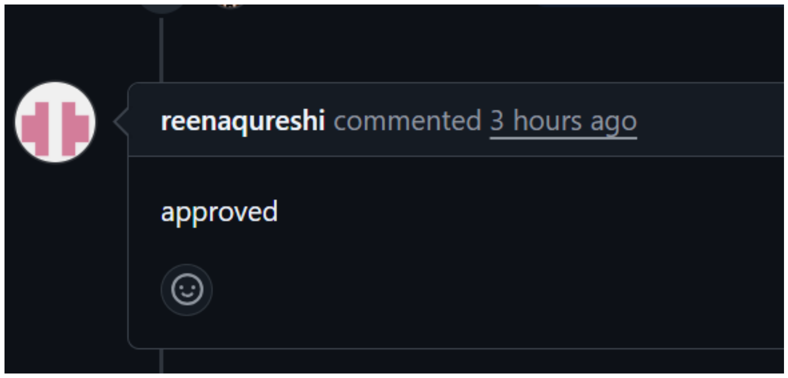

Manage access

Reena Qureshi

reenaqureshi • Collaborator

Reviewer Actions (capture screenshots for each):

Approve the PR

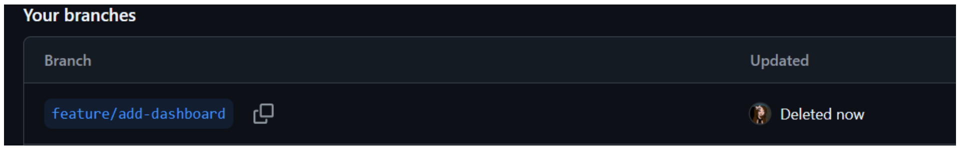

reenaqureshi commented 3 hours ago

approved

Task 13: Branch Cleanup Best Practices

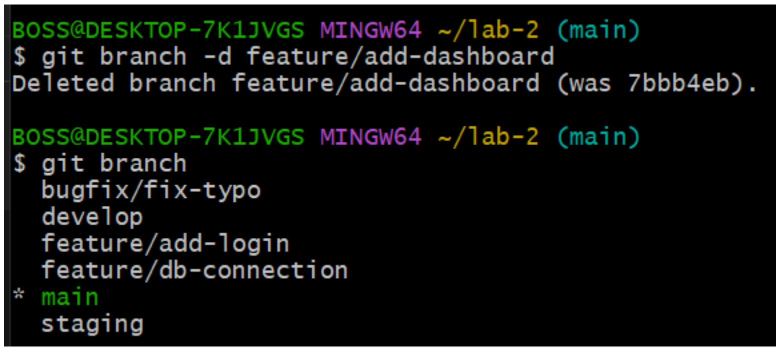
1. Delete Remote Branch After Merge:

Use GitHub UI



2. Update Local Repository:

```
git checkout main

git pull origin main

git branch -d <branch-name>

git branch
```

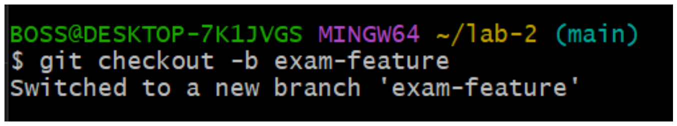


```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git branch -d feature/add-dashboard
Deleted branch feature/add-dashboard (was 7bbb4eb).

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git branch
  bugfix/fix-typo
  develop
  feature/add-login
  feature/db-connection
* main
  staging
```

EXAM EVALUATION QUESTIONS:

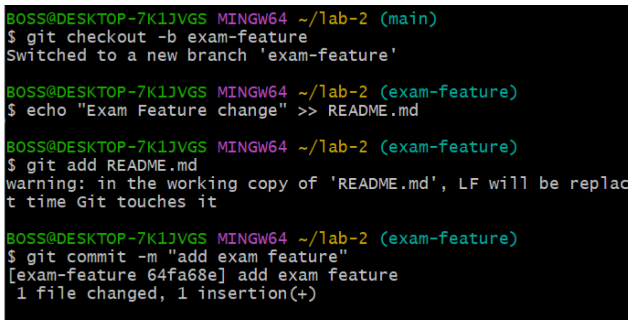
Q 1: Advanced Branching & Merge Verification

Create a new branch in your repository.



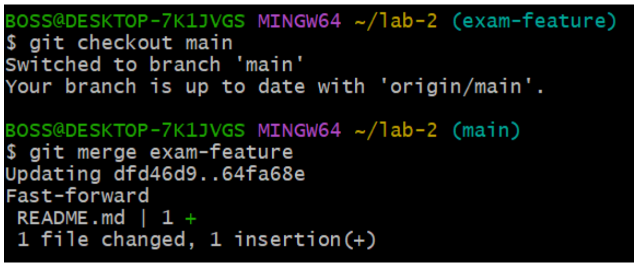
```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git checkout -b exam-feature
Switched to a new branch 'exam-feature'
```

Make a small change in a file and commit it.



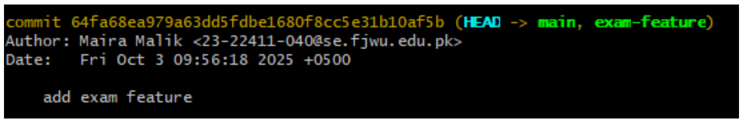
```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git checkout -b exam-feature
Switched to a new branch 'exam-feature'

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (exam-feature)
$ echo "Exam Feature change" >> README.md

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (exam-feature)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replac
t time Git touches it

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (exam-feature)
$ git commit -m "add exam feature"
[exam-feature 64fa68e] add exam feature
 1 file changed, 1 insertion(+)
```

Merge this branch back into the main branch.



```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (exam-feature)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git merge exam-feature
Updating dfd46d9..64fa68e
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
```

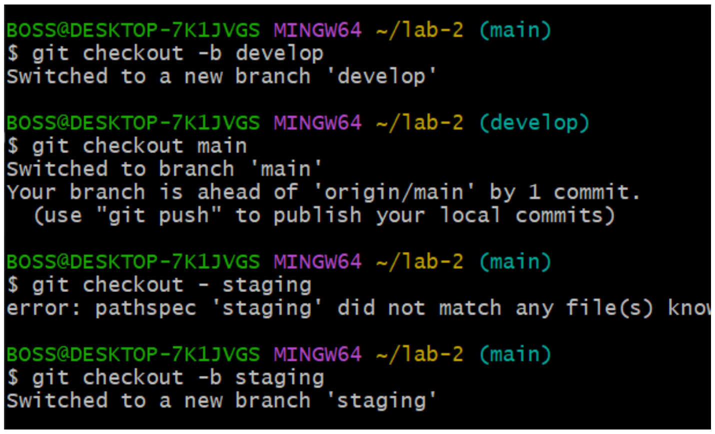
Show the history of commits in a way that verifies the merge.



```
commit 64fa68ea979a63dd5fdbe1680f8cc5e31b10af5b (HEAD -> main, exam-feature)
Author: Maira Malik <23-22411-040@se.fjwu.edu.pk>
Date:   Fri Oct 3 09:56:18 2025 +0500

    add exam feature
```

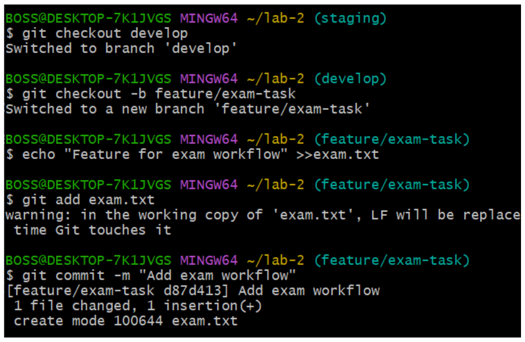
Q 2: Multi-Stage Workflow Simulation

Set up a branching workflow with three branches: `main`, `develop`, and `staging`.



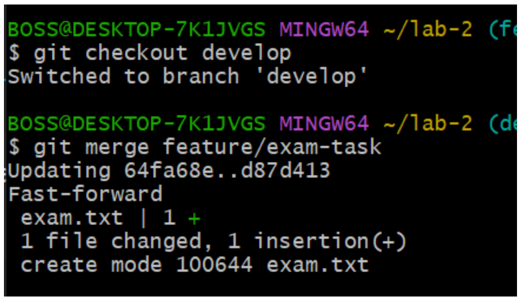
```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git checkout - staging
error: pathspec 'staging' did not match any file(s) know

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git checkout -b staging
Switched to a new branch 'staging'
```

Create a feature branch from `develop`, make changes, and commit them.



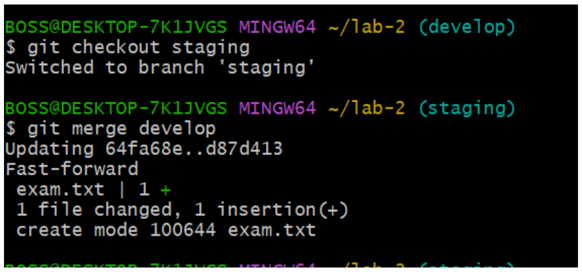
```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (staging)
$ git checkout develop
Switched to branch 'develop'

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (develop)
$ git checkout -b feature/exam-task
Switched to a new branch 'feature/exam-task'

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (feature/exam-task)
$ echo "Feature for exam workflow" >>exam.txt

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (feature/exam-task)
$ git add exam.txt
warning: in the working copy of 'exam.txt', LF will be replace
 time Git touches it

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (feature/exam-task)
$ git commit -m "Add exam workflow"
[feature/exam-task d87d413] Add exam workflow
 1 file changed, 1 insertion(+)
 create mode 100644 exam.txt
```

Merge the feature branch into `develop`.



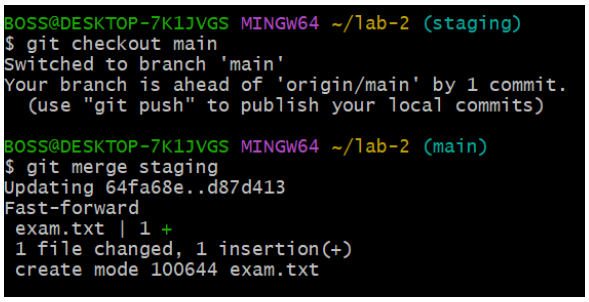
```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (fe
$ git checkout develop
Switched to branch 'develop'

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (de
$ git merge feature/exam-task
Updating 64fa68e..d87d413
Fast-forward
 exam.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 exam.txt
```

Merge `develop` into `staging`.



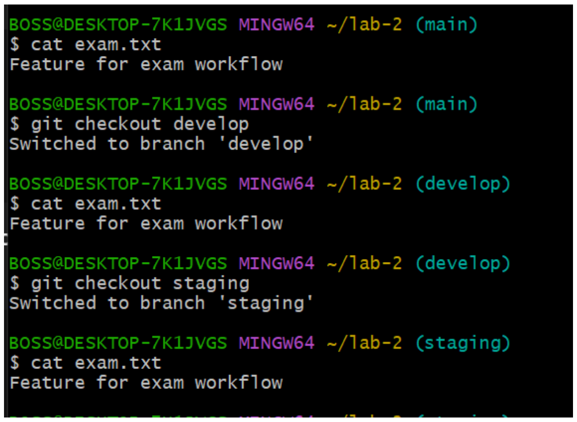
```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (develop)
$ git checkout staging
Switched to branch 'staging'

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (staging)
$ git merge develop
Updating 64fa68e..d87d413
Fast-forward
 exam.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 exam.txt
```

Merge `staging` into `main`.



```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (staging)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git merge staging
Updating 64fa68e..d87d413
Fast-forward
 exam.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 exam.txt
```

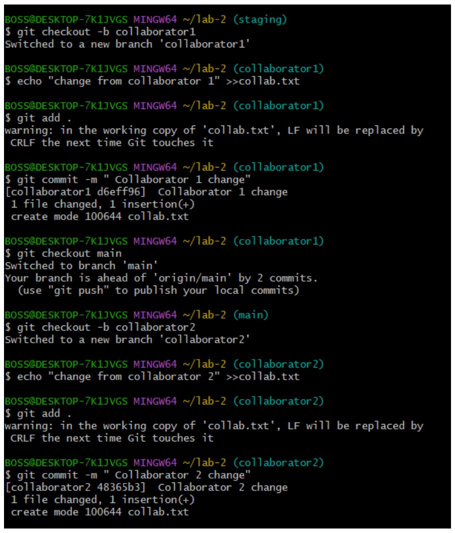
Provide proof that each stage contains the updated changes before it reaches `main`.



```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ cat exam.txt
Feature for exam workflow

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git checkout develop
Switched to branch 'develop'

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (develop)
$ cat exam.txt
Feature for exam workflow

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (develop)
$ git checkout staging
Switched to branch 'staging'

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (staging)
$ cat exam.txt
Feature for exam workflow
```

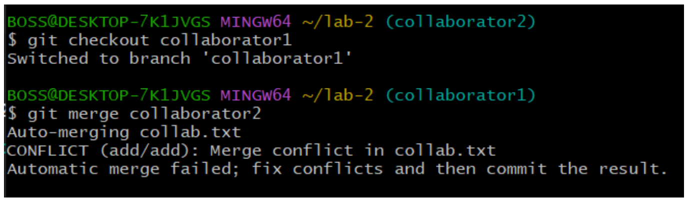
Q 3: Collaboration & Conflict Resolution

Work with a collaborator: both contributors should modify the same file but in separate branches.



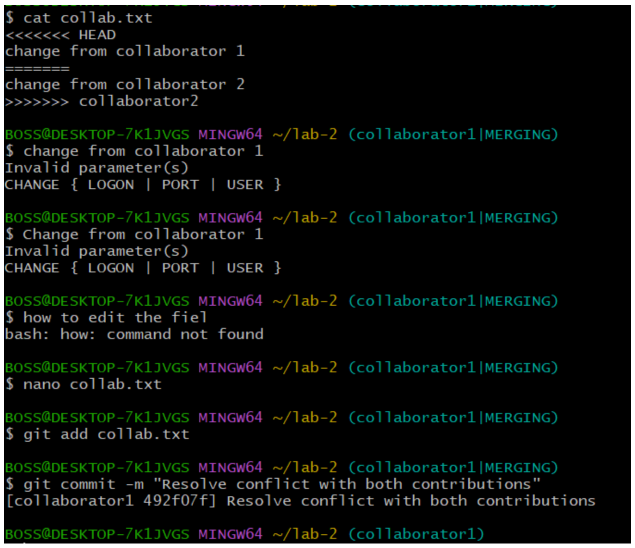
```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (staging)
$ git checkout -b collaborator1
Switched to a new branch 'collaborator1'

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1)
$ echo "change from collaborator 1" >>collab.txt

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1)
$ git add .
warning: in the working copy of 'collab.txt', LF will be replaced by
 CRLF the next time Git touches it

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1)
$ git commit -m " Collaborator 1 change"
[collaborator1 d6eff96]  Collaborator 1 change
 1 file changed, 1 insertion(+)
 create mode 100644 collab.txt

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (main)
$ git checkout -b collaborator2
Switched to a new branch 'collaborator2'

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator2)
$ echo "change from collaborator 2" >>collab.txt

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator2)
$ git add .
warning: in the working copy of 'collab.txt', LF will be replaced by
 CRLF the next time Git touches it

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator2)
$ git commit -m " Collaborator 2 change"
[collaborator2 48365b3]  Collaborator 2 change
 1 file changed, 1 insertion(+)
 create mode 100644 collab.txt
```

Attempt to merge the branches and capture the conflict.



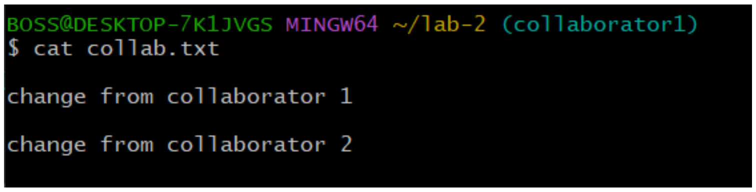
```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator2)
$ git checkout collaborator1
Switched to branch 'collaborator1'

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1)
$ git merge collaborator2
Auto-merging collab.txt
CONFLICT (add/add): Merge conflict in collab.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Resolve the conflict so that both contributions are preserved.



```
$ cat collab.txt
<<<<<<< HEAD
change from collaborator 1
=======
change from collaborator 2
>>>>>>> collaborator2

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1|MERGING)
$ change from collaborator 1
Invalid parameter(s)
CHANGE { LOGON | PORT | USER }

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1|MERGING)
$ Change from collaborator 1
Invalid parameter(s)
CHANGE { LOGON | PORT | USER }

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1|MERGING)
$ how to edit the fiel
bash: how: command not found

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1|MERGING)
$ nano collab.txt

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1|MERGING)
$ git add collab.txt

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1|MERGING)
$ git commit -m "Resolve conflict with both contributions"
[collaborator1 492f07f] Resolve conflict with both contributions

BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1)
```

Provide evidence that the final version of the file contains both collaborators' changes.



```
BOSS@DESKTOP-7K1JVGS MINGW64 ~/lab-2 (collaborator1)
$ cat collab.txt

change from collaborator 1

change from collaborator 2
```