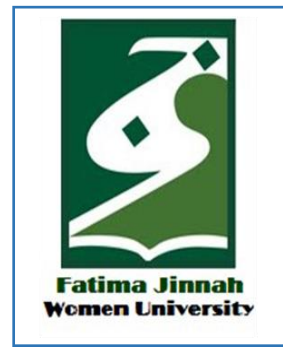


Name: Lujain Zia
Roll no: 2023-BSE-034
Date: October 16, 2025



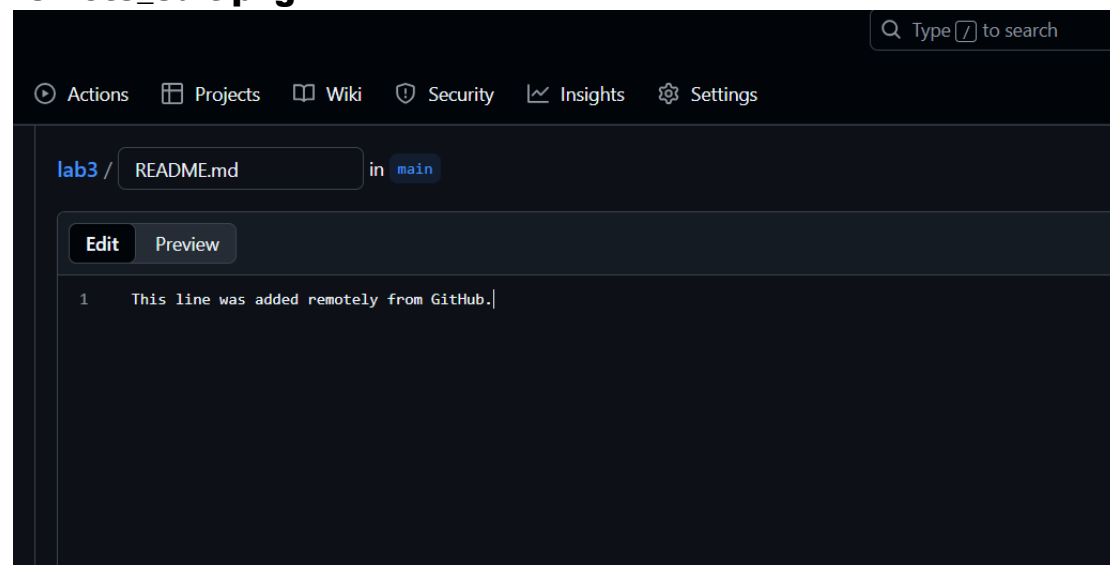
Lab 3

Task 1 – Handling Local and Remote Commit Conflicts (Pull vs Pull --rebase)

1. Open your repository on GitHub and edit the README.md file directly in the browser.
 - Add a new line such as:
This line was added remotely from GitHub.

Commit your change on GitHub (e.g., message: Remote update to README).

Save a screenshot of your browser showing the change as remote_edit.png.



2. On your local machine, open the same repository folder.

- Edit the same README.md file locally, for example:
This line was added locally.
- Stage and commit your change:
Save a screenshot of your terminal as local_commit.png.

```
MINGW64:/c/Users/user/lab3
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ nano README.md
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git add README.md
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "local update"
[main 8f83cb2] local update
1 file changed, 2 insertions(+)
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |
```

3. Try to push:

● git push origin main

You'll see an error message similar to:

! [rejected] main -> main (fetch first)

error: failed to push some refs to 'github.com:username/repo.git'

hint: Updates were rejected because the remote contains work that you do not have locally.

Save a screenshot of this error as push_error.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
To https://github.com/lujainzia127/lab3.git
! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/lujainzia127/lab3.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
```

4. To fix this, pull the latest changes from remote:

git pull --no-rebase origin main

Git will merge your local and remote commits, creating a merge commit.

Save a screenshot as merge_commit.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git pull --no-rebase origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 939 bytes | 62.00 KiB/s, done.
From https://github.com/lujainzia127/lab3
* branch                main      -> FETCH_HEAD
   8f83cb2..6a066b8      main      -> origin/main
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main|MERGING)
$
```

5. Push again:

Save a screenshot as `push_after_merge.png`.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push -u origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 656 bytes | 328.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/lujainzial27/lab3.git
   6a066b8..56bc61d  main -> main
branch 'main' set up to track 'origin/main'.

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |
```

6. Now repeat the same process, but this time use rebase instead of merge:

- Make another remote edit on GitHub in README.md.
- Then make another local edit and commit it.
- Instead of using `git pull`, run:

Save a screenshot as `rebase_pull.png`.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git pull --rebase origin main
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 9 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (9/9), 2.75 KiB | 70.00 KiB/s, done.
From https://github.com/lujainzial27/lab3
 * branch            main      -> FETCH_HEAD
   56bc61d..9e30178  main      -> origin/main
Auto-merging README.md
```

- Push again:

Save a screenshot as `push_after_rebase.png`.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main|REBASE 1/1)
$ git push -u origin main --force
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/lujainzial27/lab3.git
   + 9e30178...1b8ce7d  main -> main (forced update)
branch 'main' set up to track 'origin/main'.
```

Task 2 – Creating and Resolving Merge Conflicts Manually

1. On GitHub (remote), open your README.md file and change an existing line.

For example, if it currently says:

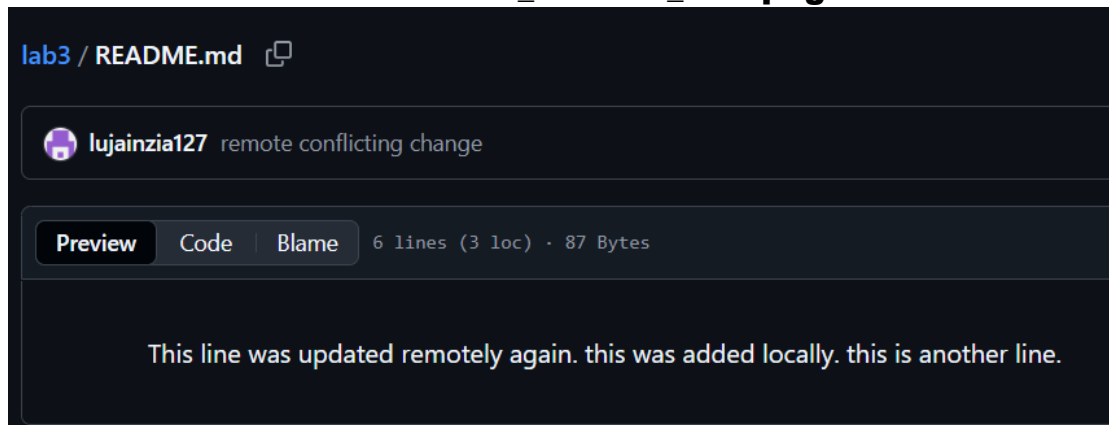
This line was added remotely from GitHub.

Change it to:

This line was updated remotely again.

Commit your change with the message: Remote conflicting change

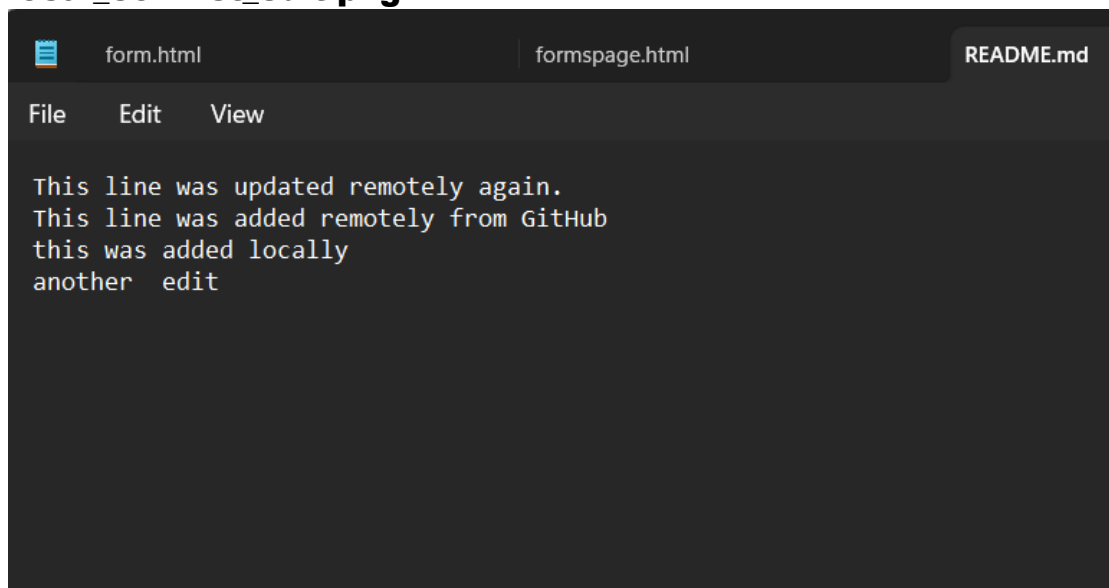
Save a screenshot as remote_conflict_edit.png.



2. On your local machine, edit the same line in the same file but make a different change:

This line was updated locally at the same time.

Save a screenshot of your edit in VS Code as local_conflict_edit.png.



3. Stage and commit your local change:

git add README.md

git commit -m "Local conflicting change"

Save a screenshot as local_conflict_commit.png.

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git add README.md

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "local conflicting change"
[main 79d5d48] local conflicting change
1 file changed, 1 insertion(+), 1 deletion(-)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$

```

4. Try to push:

The push will be rejected, since the remote version has conflicting changes.

Save a screenshot of this error as **conflict_push_error.png**.

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push -u origin main
To https://github.com/lujainzia127/lab3.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/lujainzia127/lab3.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

```

5. Pull with rebase to bring in remote changes:

git pull --rebase origin main

Git will pause the rebase and notify you about a conflict.

Save a screenshot of the conflict message as **conflict_message.png**.

```

MINGW64:/c/Users/user/lab3
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git pull --rebase origin main
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (6/6), 1.85 KiB | 17.00 KiB/s, done.
From https://github.com/lujainzia127/lab3
 * branch            main              -> FETCH_HEAD
   8d5b2e5..4ac47bf  main              -> origin/main
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
error: could not apply 145befc... change
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --
abort".
hint: Disable this message with "git config set advice.mergeConflict false"
Could not apply 145befc... # change

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main|REBASE 1/1)

```

6. Open the README.md file in your editor — you'll see conflict markers like this:

<<<<<< HEAD

This line was updated locally at the same time.

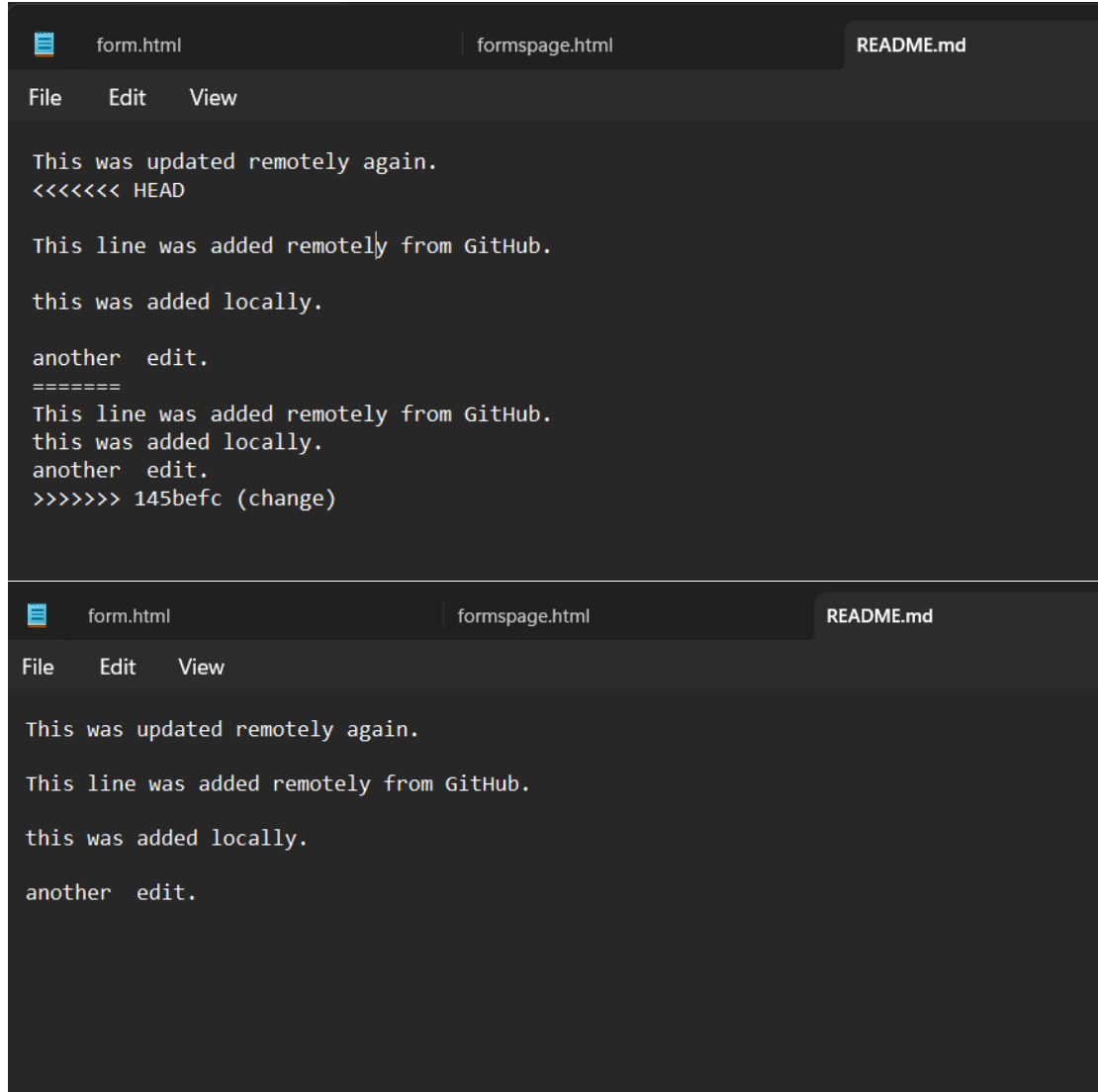
=====

This line was updated remotely again.

>>>>>> origin/main

Manually edit the file to keep the correct line (for example, merge both ideas).

Save a screenshot of the resolved file as resolved_readme.png.

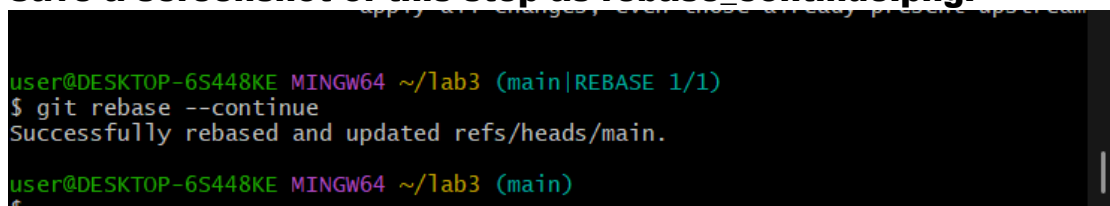


7. After fixing the file, mark the conflict as resolved and continue the rebase:

git add README.md

git rebase --continue

Save a screenshot of this step as rebase_continue.png.



8. Finally, push your changes:

git push -u origin main

Save a screenshot as push_after_resolve.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push -u origin main
branch 'main' set up to track 'origin/main'.
Everything up-to-date

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
Everything up-to-date
```

Task 3 – Managing Ignored Files with .gitignore and Removing Tracked Files

Steps

1. Create a new folder named textfiles inside your repository:

mkdir textfiles

2. Inside the textfiles folder, create three text files:

echo "File A content" > textfiles/a.txt

echo "File B content" > textfiles/b.txt

echo "File C content" > textfiles/c.txt

3. Add and commit the new directory:

git add .

git commit -m "Added textfiles directory with three files"

git push origin main

Save a screenshot as push_textfiles.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "added files"
[main 1f30448] added files
3 files changed, 3 insertions(+)
create mode 100644 textfiles/a.txt
create mode 100644 textfiles/b.txt
create mode 100644 textfiles/c.txt

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 435 bytes | 54.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/lujainzia127/lab3.git
4ac47bf..1f30448 main -> main

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |
```

4. Now, create a .gitignore file in the root of your repository:

echo "textfiles/*" > .gitignore

5. Add and commit the .gitignore file:

git add .gitignore

git commit -m "Added .gitignore to ignore textfiles directory"

git push origin main

Save a screenshot as gitignore_push.png.

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "added .gitignore"
[main 8d75d7e] added .gitignore
1 file changed, 1 insertion(+)
create mode 100644 .gitignore

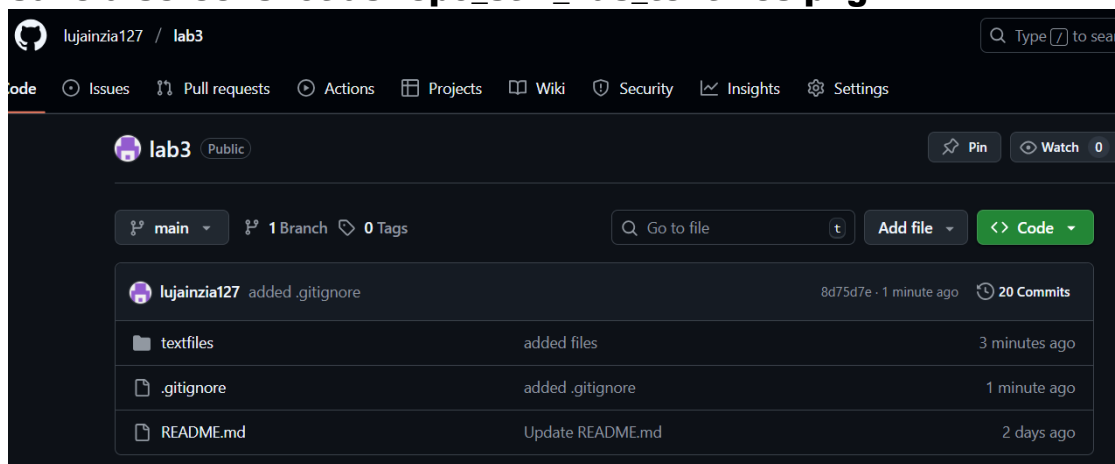
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 327 bytes | 109.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/lujainzia127/lab3.git
1f30448..8d75d7e  main -> main

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |

```

6. Go to GitHub and check your repository — notice that the textfiles directory is still visible on the remote.

Save a screenshot as repo_still_has_textfiles.png.



7. To remove already tracked files from Git (but not from your local system), run:

```
git rm -r --cached textfiles
```

8. Commit and push again:

```
git add .
```

```
git commit -m "Removed tracked textfiles directory"
```

```
git push origin main
```

Save a screenshot as rm_cached_push.png.

```

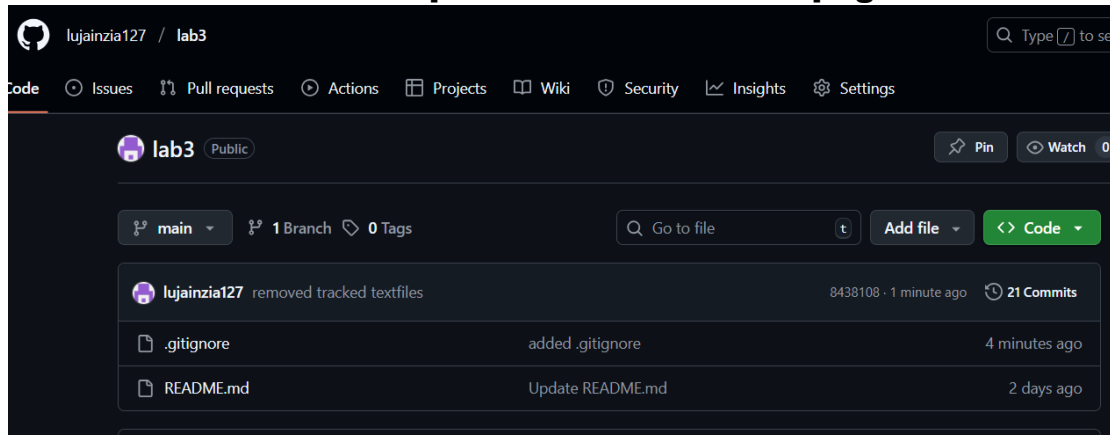
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 229 bytes | 229.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/lujainzia127/lab3.git
8d75d7e..8438108  main -> main

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |

```


9. Check your GitHub repository again — the textfiles folder should now be deleted remotely.

Save a screenshot as repo_textfiles_removed.png.



Task 4 – Create Temporary Changes and Use git stash

1. Create a feature-branch.

`git checkout -b feature-branch`

2. Modify any file (for example, README.md) by adding a few test lines.

`git add README.md`

`git commit -m "Changes the README.md file"`

Save a screenshot as modified_readme.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git checkout -b feature
Switched to a new branch 'feature'

user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$ touch README.md

user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$ git add README.md

user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$ git commit -m "changes made"
[feature 32571a7] changes made
1 file changed, 2 insertions(+)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$
```

3. Modify any file (for example, README.md) by adding a few new lines.

4. Without committing or stashing the changes, try to switch to another branch:

`git checkout main`

You'll see an error message similar to:

error: Your local changes to the following files would be overwritten by checkout:

README.md

Please commit your changes or stash them before you switch branches.

Save a screenshot as checkout_error.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$ git commit -m "changes made"
[feature 32571a7] changes made
1 file changed, 2 insertions(+)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$ git checkout main
error: Your local changes to the following files would be overwritten by checkout:
    README.md
Please commit your changes or stash them before you switch branches.
Aborting
```

5. To fix this, stash your changes:

git stash

Save a screenshot as stash_command.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$ git stash
Saved working directory and index state WIP on feature: 32571a7 changes made
```

6. Try switching branches again — it should now work:

git checkout main

Save a screenshot as branch_switched.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$
```

7. Return to your previous branch (for example, feature-branch):

git checkout feature-branch

Save a screenshot as back_to_feature.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git checkout feature
Switched to branch 'feature'

user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$
```

8. Check your working directory status:

git status

It should show:

nothing to commit, working tree clean

Save a screenshot as status_clean.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$ git status
On branch feature
nothing to commit, working tree clean

user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$
```

9. Now restore your stashed changes:

git stash pop

You'll see a message indicating that changes were reapplied:

On branch feature-branch

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)

Save a screenshot as stash_pop.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$ git stash pop
On branch feature
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")
Dropped refs/stash@{0} (9925c1e2acc90b4f7839301e3159634576116caa)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$
```

10. Confirm that your previous edits are back in the file.

```
new change
these are temporary changes|
```

Task 5 – Checkout a Specific Commit Using git log

1. View commit history:

git log --oneline

Save a screenshot as log_before_checkout.png.

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$ git log --oneline
32571a7 (HEAD -> feature) changes made
8438108 (origin/main, origin/HEAD, main) removed tracked textfiles
8d75d7e added .gitignore
1f30448 added files
4ac47bf Update README.md
86c7847 Update README.md
8d5b2e5 local conflicting
79d5d48 local conflicting change
104706c Update README.md
5bd5d9d Update README.md
c3d87b6 Update README.md
3df68eb Update README.md
89ae0eb remote conflicting change
866be2b Update README.md
4c81fea Update README.md
1b8ce7d another edit
56bc61d changes
050fe15 local update
6a066b8 Update README.md
8f83cb2 local update
8796472 Update README.md
18909c7 Create README.md

user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)

```

2. Copy any previous commit hash (e.g., a1b2c3d).

3. Checkout that commit:

`git checkout <commit-hash>`

Save a screenshot as detached_head.png.

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (feature)
$ git checkout 104706c
Note: switching to '104706c'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 104706c Update README.md

user@DESKTOP-6S448KE MINGW64 ~/lab3 ((104706c...))

```

4. To return to your main branch:

`git checkout main`

Save a screenshot as back_to_main.png.

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 ((104706c...))
$ git checkout main
Previous HEAD position was 104706c Update README.md
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)

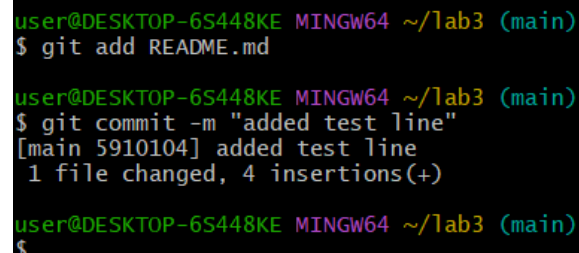
```

Task 6 – Resetting Commits (Soft vs Hard Reset) (With Verification Steps)Steps

1. Add a new line in any file and commit it:

Edit any file (e.g., README.md), add a line
git add .
git commit -m "Added test line"

Save a screenshot as first_commit.png.

A terminal window with a dark background and light green text. The prompt is 'user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)'. The first command is '\$ git add README.md'. The second prompt is '\$ git commit -m "added test line"'. The output is '[main 5910104] added test line' followed by '1 file changed, 4 insertions(+)' on the next line. The third prompt is '\$' and the cursor is on a new line.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git add README.md

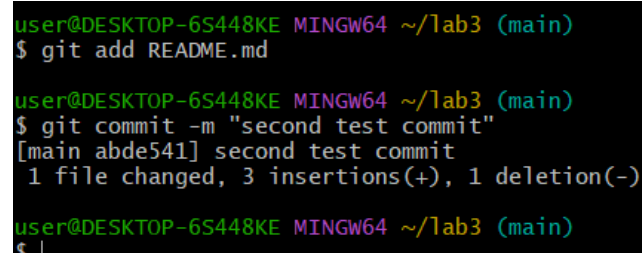
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "added test line"
[main 5910104] added test line
1 file changed, 4 insertions(+)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$
```

2. Add another change and commit again:

Edit the file again, add a different line
git add .
git commit -m "Second test commit"

Save a screenshot as second_commit.png.

A terminal window with a dark background and light green text. The prompt is 'user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)'. The first command is '\$ git add README.md'. The second prompt is '\$ git commit -m "second test commit"'. The output is '[main abde541] second test commit' followed by '1 file changed, 3 insertions(+), 1 deletion(-)' on the next line. The third prompt is '\$' and the cursor is on a new line.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git add README.md

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "second test commit"
[main abde541] second test commit
1 file changed, 3 insertions(+), 1 deletion(-)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$
```

3. View the history before reset:

git log --oneline

Save a screenshot as log_before_reset.png.

```
$ git log --oneline
abde541 (HEAD -> main) second test commit
5910104 added test line
8438108 (origin/main, origin/HEAD) removed tracked textfiles
8d75d7e added .gitignore
1f30448 added files
4ac47bf Update README.md
86c7847 Update README.md
8d5b2e5 local conflicting
79d5d48 local conflicting change
104706c Update README.md
5bd5d9d Update README.md
c3d87b6 Update README.md
3df68eb Update README.md
89ae0eb remote conflicting change
866be2b Update README.md
4c81fea Update README.md
1b8ce7d another edit
56bc61d changes
050fe15 local update
6a066b8 Update README.md
8f83cb2 local update
8796472 Update README.md
18909c7 Create README.md
```

4. Check file contents for both changes:

Open the edited file (e.g., README.md) and visually confirm BOTH added lines exist.

Save a screenshot as file_before_reset.png.

```
this is first test line

this is second test line|
```

5. Perform a soft reset (keeps changes in working directory):

```
git reset --soft HEAD~1
```

Save a screenshot as soft_reset.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git reset --soft HEAD~1

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |
```

6. Check commit history after soft reset:

```
git log --oneline
```

Save a screenshot as log_after_soft_reset.png.

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git log --oneline
5910104 (HEAD -> main) added test line
8438108 (origin/main, origin/HEAD) removed tracked textfiles
8d75d7e added .gitignore
1f30448 added files
4ac47bf Update README.md
86c7847 Update README.md
8d5b2e5 local conflicting
79d5d48 local conflicting change
104706c Update README.md
5bd5d9d Update README.md
c3d87b6 Update README.md
3df68eb Update README.md
89ae0eb remote conflicting change
866be2b Update README.md
4c81fea Update README.md
1b8ce7d another edit
56bc61d changes
050fe15 local update
6a066b8 Update README.md
8f83cb2 local update
8796472 Update README.md
18909c7 Create README.md

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)

```

7. Verify changes in the file:

Open the file again and confirm both edits are still present (nothing lost).

Save a screenshot as file_after_soft_reset.png.

```

this is first test line

this is second test line

```

8. Check git status:

git status

You should see your changes are staged (ready to commit again).

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

```

9. Perform commit

git commit -m "Second Test commit"

Save a screenshot as file_after_hard_reset.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "second test commit"
[main 1e2ca1f] second test commit
1 file changed, 3 insertions(+), 1 deletion(-)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$
```

10. Perform a hard reset (discards changes completely):

git reset --hard HEAD~1

Save a screenshot as hard_reset.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git reset --hard HEAD~1
HEAD is now at 5910104 added test line

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$
```

11. Check commit history after hard reset:

git log --oneline

Save a screenshot as log_after_hard_reset.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git log --oneline
5910104 (HEAD -> main) added test line
8438108 (origin/main, origin/HEAD) removed tracked textfiles
8d75d7e added .gitignore
1f30448 added files
4ac47bf Update README.md
86c7847 Update README.md
8d5b2e5 local conflicting
79d5d48 local conflicting change
104706c Update README.md
5bd5d9d Update README.md
c3d87b6 Update README.md
3df68eb Update README.md
89ae0eb remote conflicting change
866be2b Update README.md
4c81fea Update README.md
1b8ce7d another edit
56bc61d changes
050fe15 local update
6a066b8 Update README.md
8f83cb2 local update
8796472 Update README.md
18909c7 Create README.md

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |
```

12. Verify changes in the file:

Open the file and confirm the second edit is gone (reverted to previous commit).

Save a screenshot as file_after_hard_reset.png.


```
this is first test line|
```

13. Check git status:

git status

Should report "nothing to commit, working tree clean".

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
```

Task 7 – Amending the Last Commit

1. Make a small change in any file.

2. Stage it and commit:

git add .

git commit -m "Fix log message"

Save a screenshot as first_amend_commit.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git add .

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "fix log message"
[main e7acbc8] fix log message
1 file changed, 3 insertions(+), 1 deletion(-)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |
```

3. Realize you forgot to change another file. Update it now.

4. Add it and amend the last commit:

git add .

git commit --amend

Save a screenshot as amend_commit.png.


```
[main 665c673] Revert "added temporary text"
1 file changed, 1 deletion(-)
delete mode 100644 temp.txt
```

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$
```

3. Push the revert commit:

git push origin main

Save a screenshot as revert_push.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 256 bytes | 128.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/lujainzia127/lab3.git
fd7f078..665c673 main -> main
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$
```

Task 9 – Force Push (With Caution)

Use this only in your own branch (never on main or develop).

1. Create a new branch:

git checkout -b test-force

Save a screenshot as new_branch.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git checkout -b test-force
Switched to a new branch 'test-force'

user@DESKTOP-6S448KE MINGW64 ~/lab3 (test-force)
$
```

2. Make and commit a small change.

Save a screenshot as force_commit.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (test-force)
$ git commit -m "first commit for this branch"
[test-force 2a9fd96] first commit for this branch
1 file changed, 3 insertions(+), 1 deletion(-)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (test-force)
$
```

3. Push it:

git push origin test-force

Save a screenshot as push_force_branch.png.

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (test-force)
$ git push origin test-force
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 331 bytes | 66.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'test-force' on GitHub by visiting:
remote:   https://github.com/lujainzia127/lab3/pull/new/test-force
remote:
To https://github.com/lujainzia127/lab3.git
 * [new branch]      test-force -> test-force

user@DESKTOP-6S448KE MINGW64 ~/lab3 (test-force)
$

```

4. Perform a hard reset:

git reset --hard HEAD~1

Save a screenshot as hard_reset_force.png.

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (test-force)
$ git reset --hard HEAD~1
HEAD is now at 665c673 Revert "added temporary text"

user@DESKTOP-6S448KE MINGW64 ~/lab3 (test-force)
$

```

5. Push again

git push origin test-force

Rejected by remote repository

Save a screenshot as normal_push.png.

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (test-force)
$ git push origin test-force
To https://github.com/lujainzia127/lab3.git
 ! [rejected]        test-force -> test-force (non-fast-forward)
error: failed to push some refs to 'https://github.com/lujainzia127/lab3.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. If you want to integrate the remote changes,
hint: use 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

user@DESKTOP-6S448KE MINGW64 ~/lab3 (test-force)
$

```

6. Now force-push to remote:

git push origin test-force --force

Save a screenshot as force_push.png.

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (test-force)
$ git push origin test-force --force
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/lujainzia127/lab3.git
 + 2a9fd96...665c673 test-force -> test-force (forced update)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (test-force)
$

```

Task 10 – Running Gitea in GitHub Codespaces via Docker

Compose

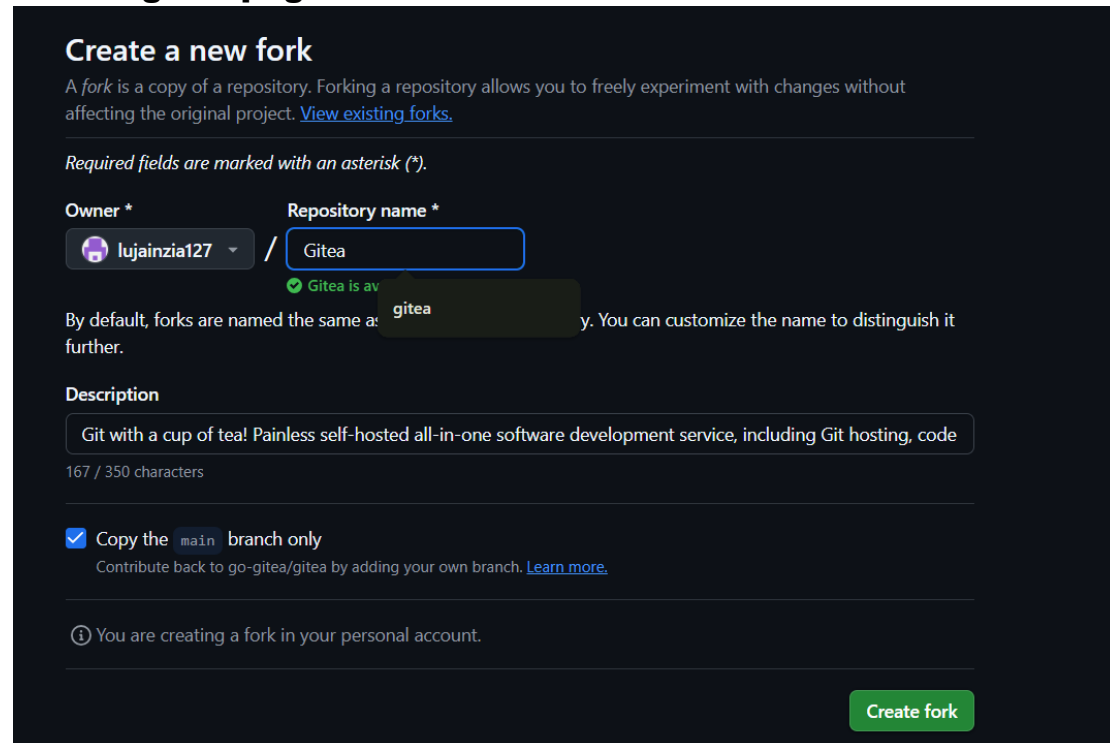
Steps

1. Fork the Gitea Repository

Go to Gitea.

Click the Fork button at the top right and fork it into your own GitHub account.

Save a screenshot of your forked repository page as `forked_gitea.png`.




Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk (*).

Owner * / Repository name *

 lujainzia127 / Gitea

✔ Gitea is available

By default, forks are named the same as the original repository. You can customize the name to distinguish it further.


Description

Git with a cup of tea! Painless self-hosted all-in-one software development service, including Git hosting, code

167 / 350 characters

☒ Copy the `main` branch only

Contribute back to go-gitea/gitea by adding your own branch. [Learn more.](#)

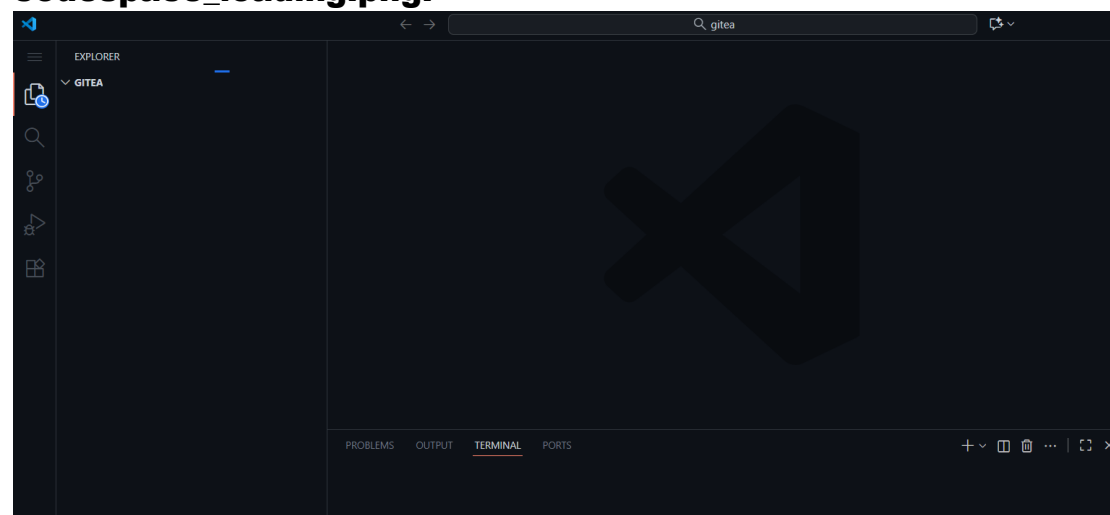
 You are creating a fork in your personal account.

[Create fork](#)

2. Open the Forked Repo in GitHub Codespaces

On your forked repo, click Code > Codespaces > Create codespace on main.

Save a screenshot of the Codespace loading as `codespace_loading.png`.



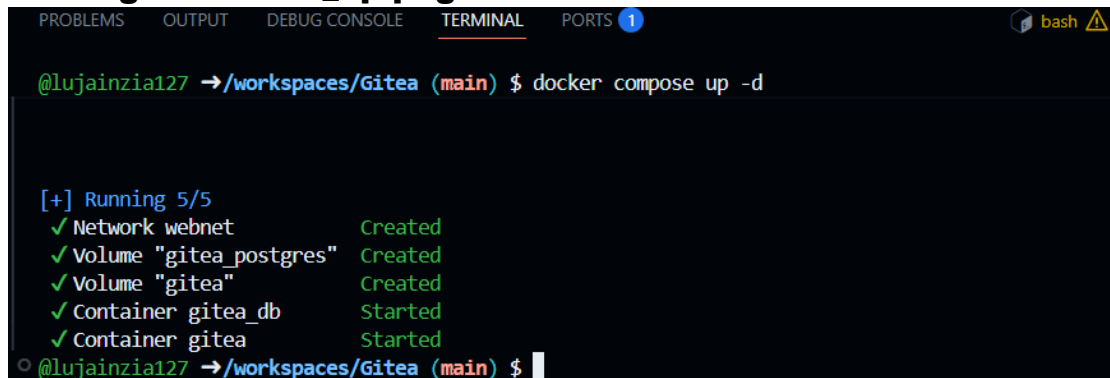
3. Start Gitea with Docker Compose

In the Codespace terminal, run:

```
docker compose up -d
```

Wait for the containers to start.

Save a screenshot of the terminal showing the containers running as docker_up.png.



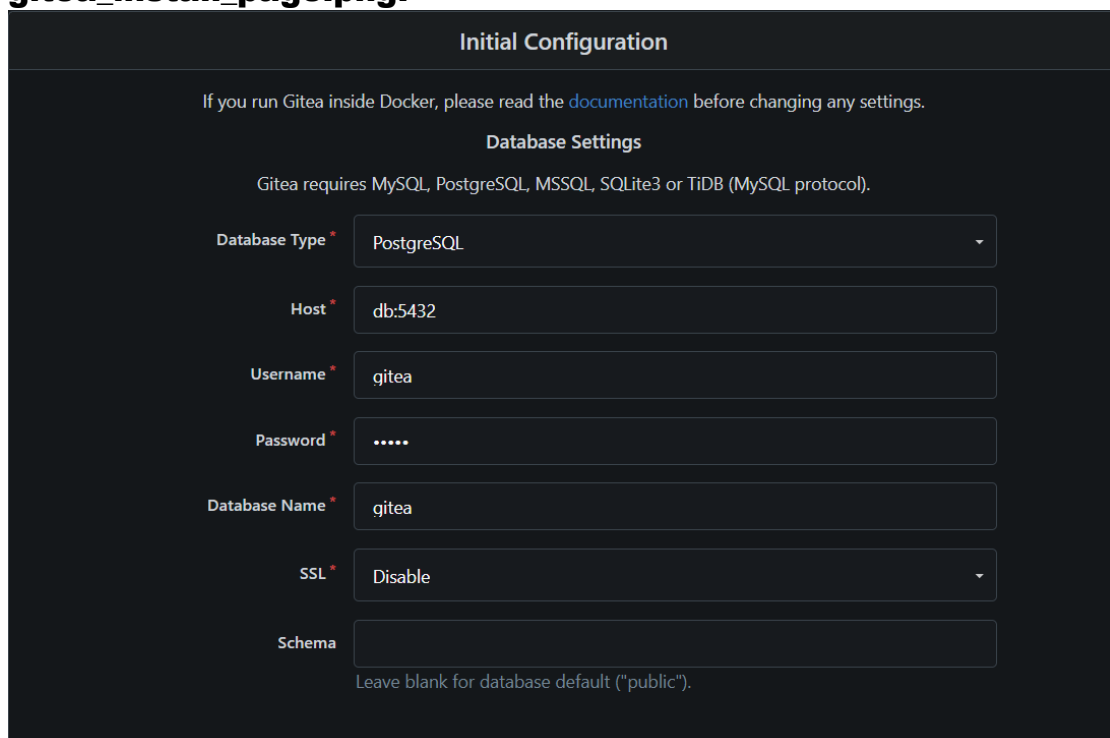
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 bash
@lujainzia127 →/workspaces/Gitea (main) $ docker compose up -d

[+] Running 5/5
  ✓ Network webnet          Created
  ✓ Volume "gitea_postgres" Created
  ✓ Volume "gitea"          Created
  ✓ Container gitea_db      Started
  ✓ Container gitea         Started
@lujainzia127 →/workspaces/Gitea (main) $
```

4. Access Gitea Web Interface

In Codespaces, forward port 3000 (see Codespaces port forwarding UI). Change the Visibility of port 3000 from private to public. Click the forwarded port 3000 link to open Gitea in your browser tab.

Save a screenshot of the Gitea install page as gitea_install_page.png.



Initial Configuration

If you run Gitea inside Docker, please read the [documentation](#) before changing any settings.

Database Settings

Gitea requires MySQL, PostgreSQL, MSSQL, SQLite3 or TiDB (MySQL protocol).

Database Type *

Host *

Username *

Password *

Database Name *

SSL *

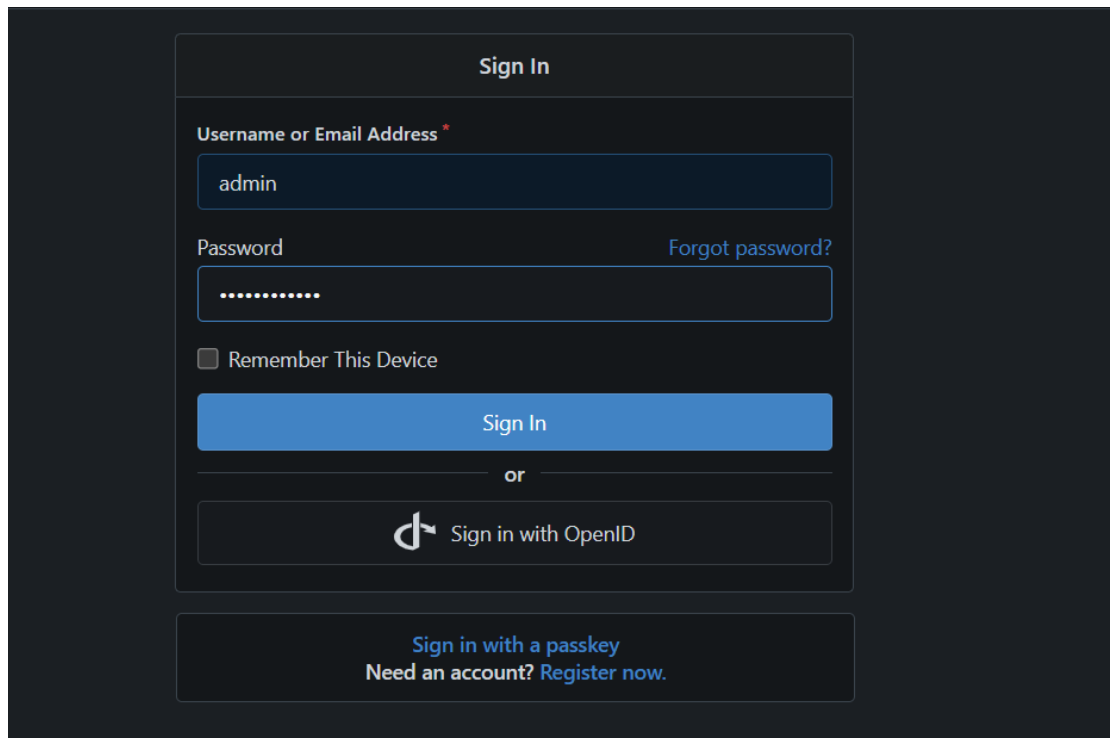
Schema

Leave blank for database default ("public").

5. Install Gitea

Fill out the installation form, providing an admin username and password (e.g., admin / yourpassword).

Save a screenshot of the completed setup (show the admin account setup) as admin_setup.png.



The image shows the Gitea 'Sign In' interface. It features a dark theme with a central white box containing the login fields. The 'Username or Email Address' field is filled with 'admin'. The 'Password' field is masked with dots. A 'Remember This Device' checkbox is present but unchecked. Below the fields is a large blue 'Sign In' button. Underneath this button is a horizontal line with the word 'or' in the center. Below the line is a button with the OpenID icon and the text 'Sign in with OpenID'. At the bottom of the form is a link 'Sign in with a passkey' and another link 'Need an account? Register now.'.

Sign In

Username or Email Address *

admin


Password [Forgot password?](#)

.....

☐ Remember This Device

Sign In

or

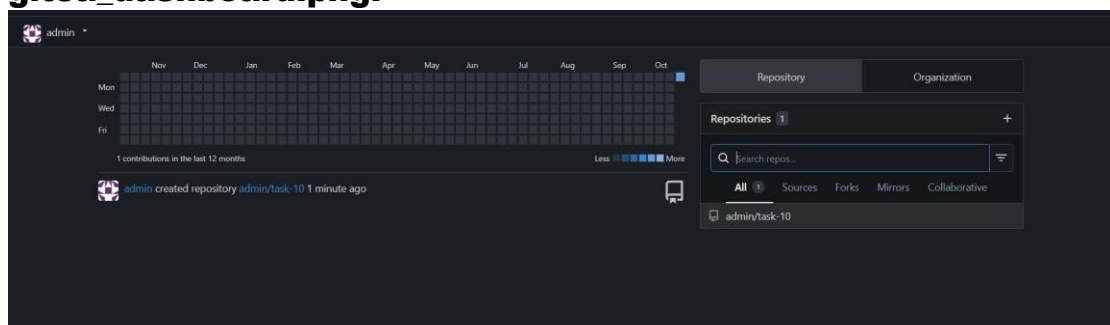
 Sign in with OpenID

[Sign in with a passkey](#)
[Need an account? Register now.](#)

6. Log In to Gitea

Use your admin credentials to log in.

Save a screenshot of the Gitea dashboard after login as gitea_dashboard.png.



7. Create a New Repository in Gitea


Click New Repository or + > New Repository in Gitea.

Fill out the repository details and create it.

Save a screenshot of the new repository page as gitea_new_repo.png.

New Repository

A repository contains all project files, including revision history. Already hosting one elsewhere? [Migrate repository.](#)

Owner *  admin

Some organizations may not show up in the dropdown due to a maximum repository count limit.

Repository Name *

Good repository names use short, memorable and unique keywords. A repository named ".profile" or ".profile-private" could be used to add a README.md for the user/organization profile.

Visibility ☐ Make repository private

Only the owner or the organization members if they have rights, will be able to see it.

Description

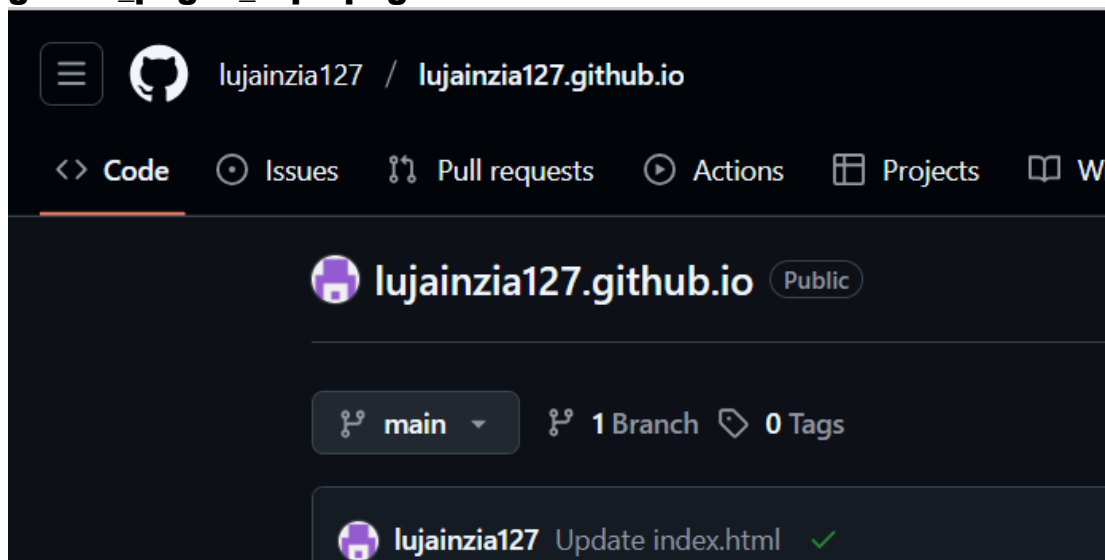
Task 11 – Creating a GitHub Pages Portfolio Site

1. Create a GitHub Pages Repository

Go to GitHub and create a new repository named <your-username>.github.io (e.g., if your username is WaqasSaleem97, the repo should be WaqasSaleem97.github.io).

Make sure the repository is public.

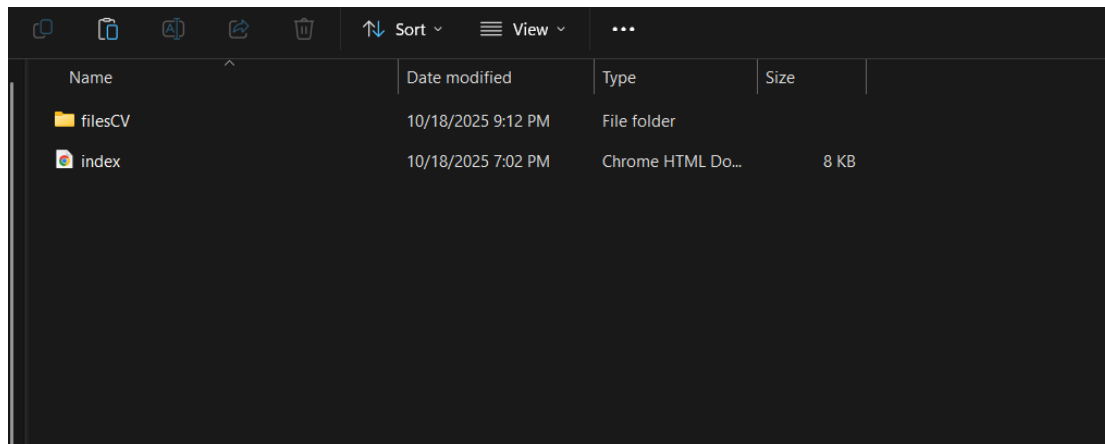
Save a screenshot of the newly created repository as github_pages_repo.png.



2. Add Static Website Code

Prepare a simple static website (e.g., your CV or portfolio) in HTML/CSS/JS by using the sample code provided in the repository lqra Bashir CV. Put your files in a local folder.

Save a screenshot of your local files as local_static_site.png.



2. Push the Files to GitHub

Initialize git in your folder (if not already):

```
git init
```

```
git remote add origin git@github.com:<your-username>/<your-username>.github.io.git
```

Add and commit your files:

```
git add .
```

```
git commit -m "Add portfolio site for GitHub Pages"
```

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

Save a screenshot of your terminal showing a successful push as push_static_site.png.

```
user@DESKTOP-6S448KE MINGW64 ~/lujainzia127.github.io (main)
$ git add .

user@DESKTOP-6S448KE MINGW64 ~/lujainzia127.github.io (main)
$ git commit -m "cv"
[main db650cf] cv
1 file changed, 194 insertions(+)
create mode 100644 filesCV/index.html

user@DESKTOP-6S448KE MINGW64 ~/lujainzia127.github.io (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 261 bytes | 43.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/lujainzia127/lujainzia127.github.io.git
13a9d4c..db650cf main -> main

user@DESKTOP-6S448KE MINGW64 ~/lujainzia127.github.io (main)
$ |
```

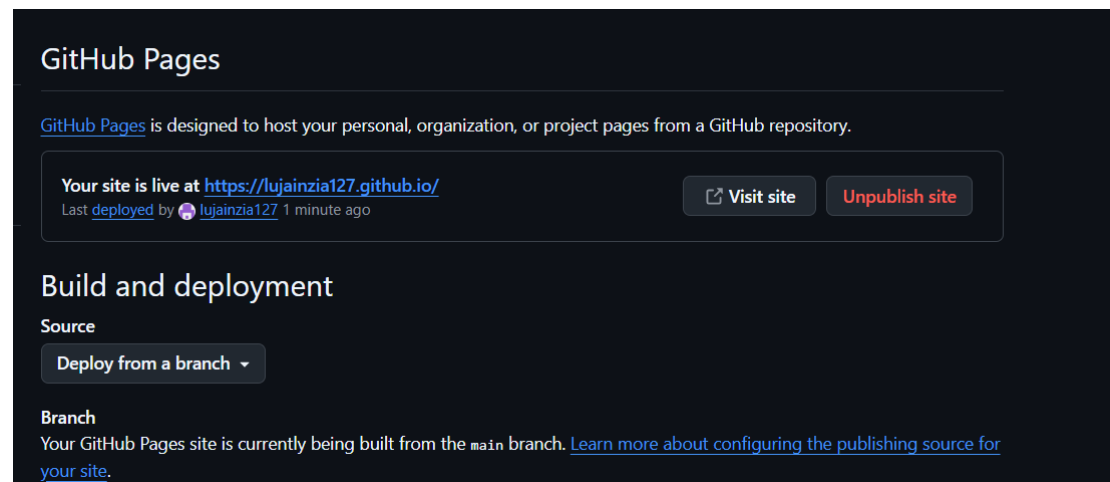
3. Check GitHub Pages Settings

Go to your repository on GitHub.

Click Settings > Pages.

Confirm that your site is published and see the link to your live site (e.g., <https://your-username.github.io>).

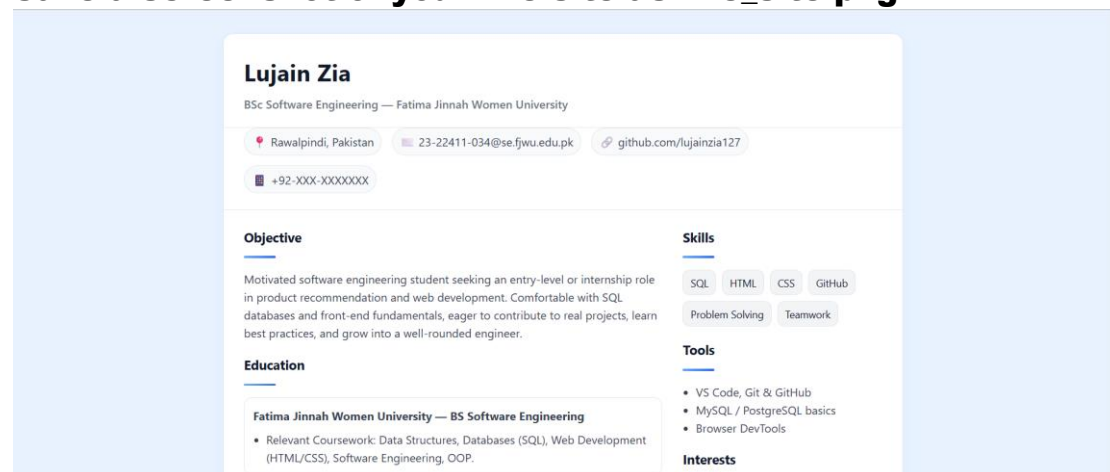
Save a screenshot of the Pages settings as github_pages_settings.png.



4. Visit Your Live Site

Open your GitHub Pages site in the browser.

Save a screenshot of your live site as live_site.png.



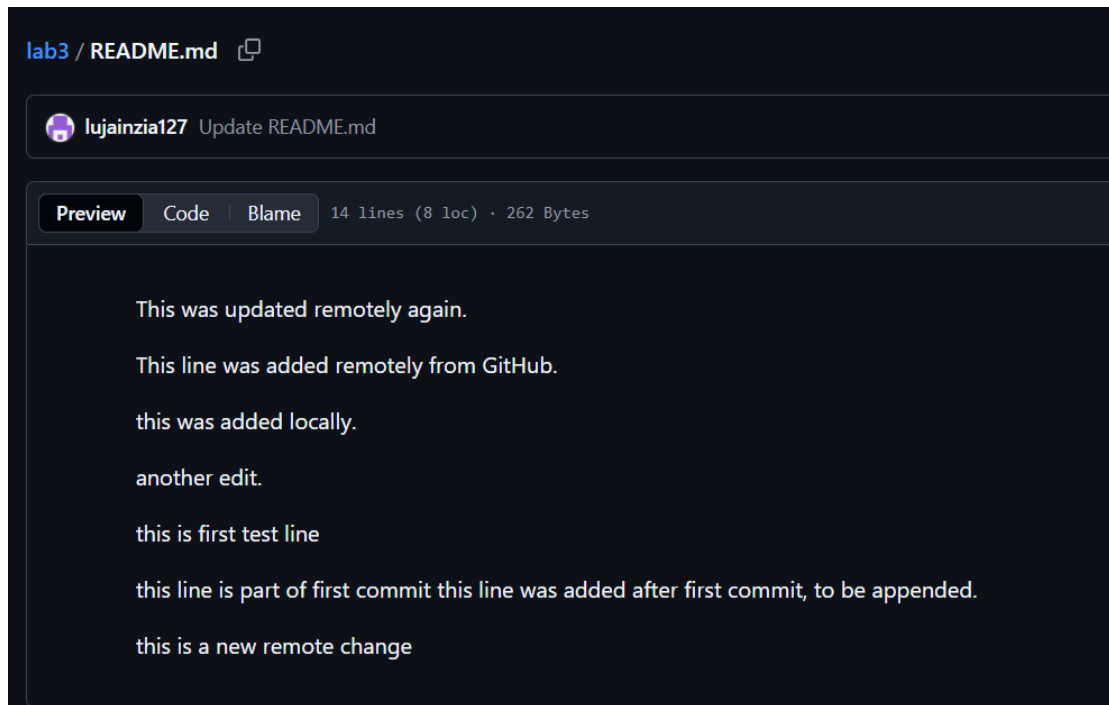
Exam Evaluation Questions

1. Local vs Remote Conflict Resolution

Steps:

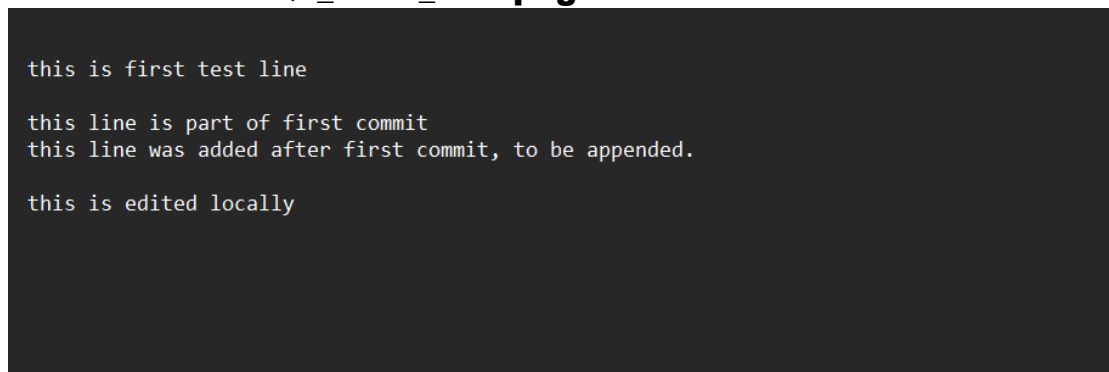
On GitHub, edit a file (e.g., README.md) and commit the change.

Screenshot as Q1_remote_edit.png



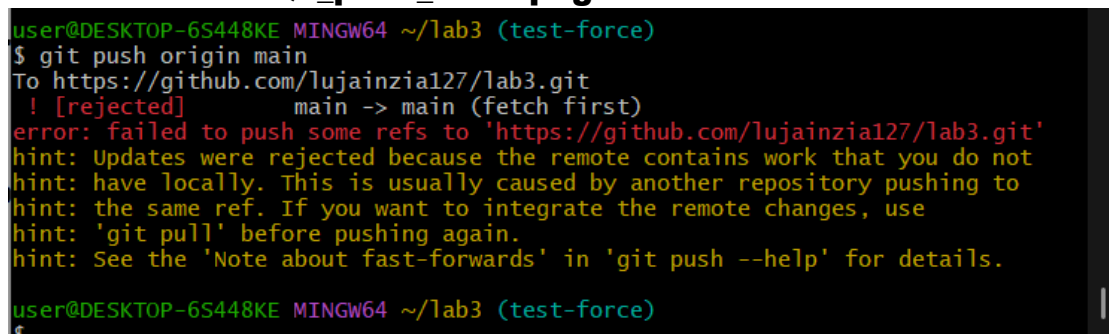
On your local machine, edit the same file differently (Avoid Conflict) and commit.

Screenshot as Q1_local_edit.png



Try to push your local commit and observe the error.

Screenshot as Q1_push_error.png



Resolve the conflict by running git pull (merge), then push.

Screenshot as Q1_merge_resolution.png

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "git mege"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

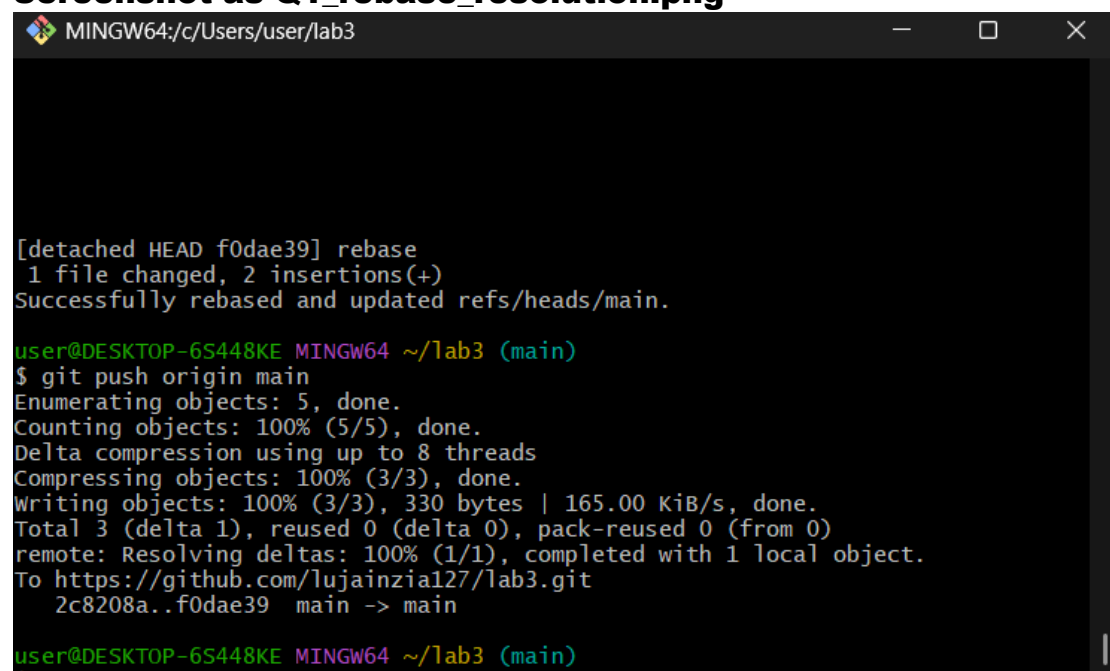
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
Everything up-to-date

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |

```

Repeat with another remote/local change, but resolve using git pull --rebase, then push.

Screenshot as Q1_rebase_resolution.png



```

MINGW64:/c/Users/user/lab3

[detached HEAD f0dae39] rebase
1 file changed, 2 insertions(+)
Successfully rebased and updated refs/heads/main.

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 330 bytes | 165.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/lujainzia127/lab3.git
   2c8208a..f0dae39  main -> main

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$

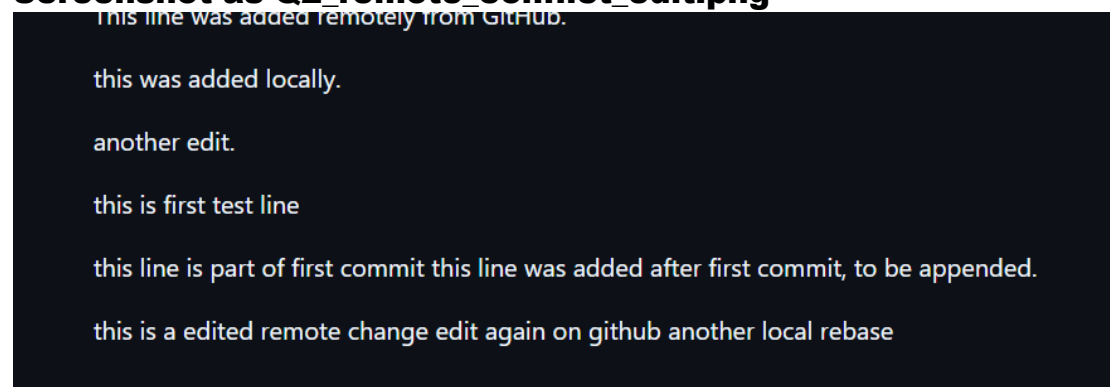
```

2. Manual Merge Conflict Handling

Steps:

On GitHub, change a specific line in a file and commit.

Screenshot as Q2_remote_conflict_edit.png



```

This line was added remotely from GitHub.

this was added locally.

another edit.

this is first test line

this line is part of first commit this line was added after first commit, to be appended.

this is a edited remote change edit again on github another local rebase

```

Locally, change the same line differently and commit.

Screenshot as Q2_local_conflict_edit.png

```
another edit.

this is first test line

this line is part of first commit
this line was added after first commit, to be appended.

this is a edited local change
edit again on github
another local rebase
```

Try to push your local change and observe the conflict error.
Screenshot as Q2_conflict_push_error.png

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
To https://github.com/lujainzia127/lab3.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/lujainzia127/lab3.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |
```

Use git pull --rebase to fetch changes and trigger the conflict.
Screenshot as Q2_rebase_conflict.png

```
MINGW64:/c/Users/user/lab3

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git pull --rebase origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 964 bytes | 80.00 KiB/s, done.
From https://github.com/lujainzia127/lab3
 * branch            main      -> FETCH_HEAD
  f0dae39..d1559b9    main      -> origin/main
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
error: could not apply eded186... Q2
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --
hint: abort".
hint: Disable this message with "git config set advice.mergeConflict false"
Could not apply eded186... # Q2

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main|REBASE 1/1)
$
```

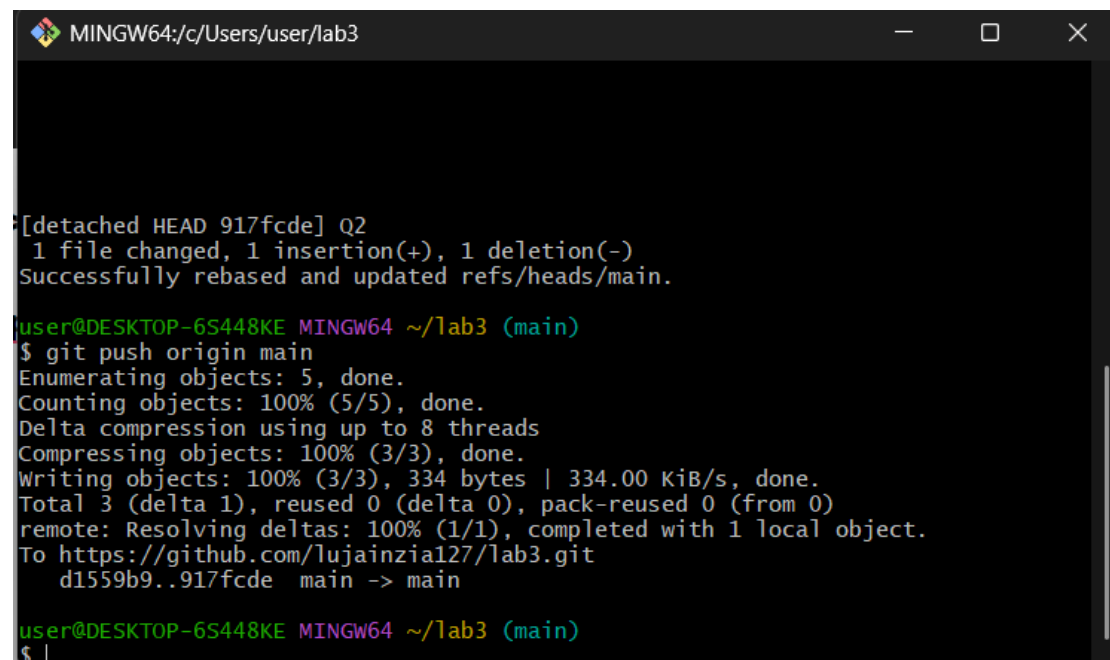
Edit the conflicted file to resolve the conflict manually.
Screenshot as Q2_resolved_file.png

```
this is first test line

this line is part of first commit
this line was added after first commit, to be appended.
this is a edited remote change
this is a edited local change
edit again on github
another local rebase
|
```

Mark the conflict resolved (git add <file>, git rebase --continue) and push.

Screenshot as Q2_resolution_complete.png

A terminal window titled 'MINGW64:/c/Users/user/lab3' showing the output of a git rebase and push. The rebase message indicates a detached HEAD, one file changed, one insertion, and one deletion, and that it was successfully rebased and updated. The push output shows the process of enumerating, counting, compressing, and writing objects, and finally pushing to the remote repository.

```
MINGW64:/c/Users/user/lab3

[detached HEAD 917fcde] Q2
1 file changed, 1 insertion(+), 1 deletion(-)
Successfully rebased and updated refs/heads/main.

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 334 bytes | 334.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/lujainzia127/lab3.git
d1559b9..917fcde  main -> main

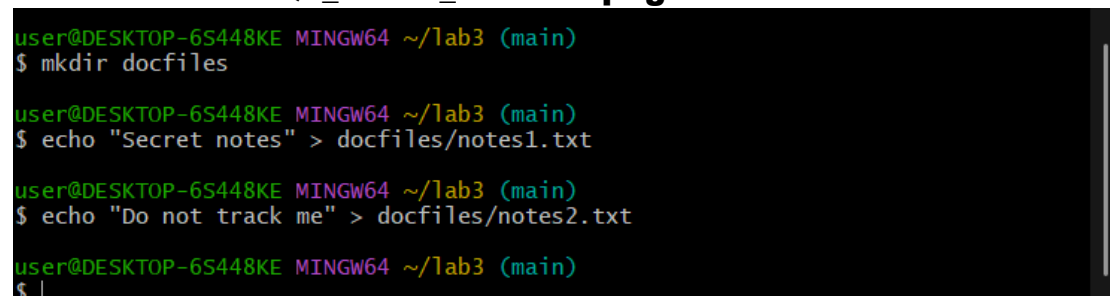
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |
```

3. Managing Ignored and Tracked Files

Steps:

Create a new folder (e.g., DocFiles) and add several files inside.

Screenshot as Q3_folder_created.png

A terminal window titled 'MINGW64:/c/Users/user/lab3' showing the creation of a 'docfiles' directory and the creation of two text files: 'notes1.txt' and 'notes2.txt'.

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ mkdir docfiles

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ echo "Secret notes" > docfiles/notes1.txt

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ echo "Do not track me" > docfiles/notes2.txt

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |
```

Commit and push the folder/files to GitHub.

Screenshot as Q3_files_pushed.png

```
MINGW64:/c/Users/user/lab3
LF the next time Git touches it
warning: in the working copy of 'docfiles/notes2.txt', LF will be replaced by CR
LF the next time Git touches it

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "add docfiles"
[main 2af74de] add docfiles
2 files changed, 2 insertions(+)
create mode 100644 docfiles/notes1.txt
create mode 100644 docfiles/notes2.txt

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 428 bytes | 214.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/lujainzia127/lab3.git
917fcde..2af74de main -> main

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$
```

Add the folder to your .gitignore file.

Screenshot as Q3_gitignore_added.png

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ echo "docfiles/" >> .gitignore

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |
```

Commit and push the .gitignore update

Screenshot as Q3_gitignore_pushed.png

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git add .gitignore
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the ne
xt time Git touches it

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "ignore docfiles"
[main 444aa6e] ignore docfiles
1 file changed, 1 insertion(+)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 334 bytes | 334.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/lujainzia127/lab3.git
2af74de..444aa6e main -> main

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |
```

Remove the folder from tracking using git rm -r --cached <folder>.

Screenshot as Q3_folder_untracked.png

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "stop tracking"
[main 3ba0eea] stop tracking
2 files changed, 2 deletions(-)
delete mode 100644 docfiles/notes1.txt
delete mode 100644 docfiles/notes2.txt

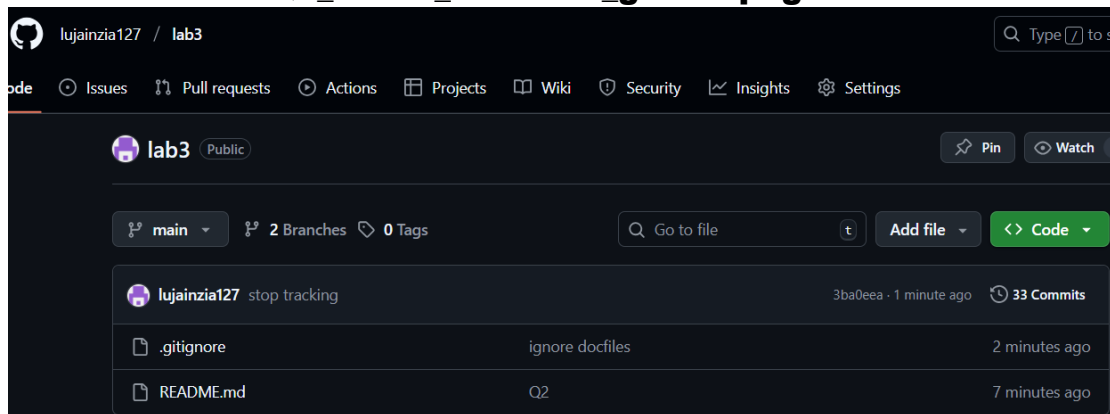
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 222 bytes | 222.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/lujainzia127/lab3.git
444aa6e..3ba0eea main -> main

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |

```

Commit and push the change, then verify the folder is no longer tracked on GitHub.

Screenshot as Q3_folder_removed_github.png



4. Commit History Manipulation and Recovery

Steps:

Make a change and commit it.

Screenshot as Q4_first_commit.png

```

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ echo "first commit line" >> test.txt

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git add test.txt
warning: in the working copy of 'test.txt', LF will be replaced by CRLF the next time Git touches it

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "first commit"
[main f585eac] first commit
1 file changed, 1 insertion(+)
create mode 100644 test.txt

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ |

```

Make another change and commit again.

Screenshot as Q4_second_commit.png


```
MINGW64:/c/Users/user/lab3
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git add test.txt
warning: in the working copy of 'test.txt', LF will be replaced by CRLF the next
time Git touches it

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "second commit"
[main 0bce7e0] second commit
1 file changed, 1 insertion(+)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
```

View your commit history.

Screenshot as Q4_commit_history.png

```
MINGW64:/c/Users/user/lab3
0bce7e0 (HEAD -> main, origin/main, origin/HEAD) second commit
f585eac first commit
3ba0eea stop tracking
444aa6e ignore docfiles
2af74de add docfiles
917fcde Q2
d1559b9 Update README.md
f0dae39 rebase
2c8208a Update README.md
df4f25c Update README.md
665c673 (origin/test-force) Revert "added temporary text"
fd7f078 added temporary text
33701d8 fix log message
5910104 added test line
8438108 removed tracked textfiles
8d75d7e added .gitignore
1f30448 added files
4ac47bf Update README.md
86c7847 Update README.md
8d5b2e5 local conflicting
79d5d48 local conflicting change
104706c Update README.md
5bd5d9d Update README.md
.
```

Perform a soft reset (git reset --soft HEAD~1) and observe your file and history.

Screenshot as Q4_soft_reset.png

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git reset --soft HEAD~1

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$
```

Make commit again.

Screenshot as Q4_third_commit.png

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git commit -m "Third commit after soft reset"
[main 17242a2] Third commit after soft reset
1 file changed, 1 insertion(+)

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$
```

Perform a hard reset (git reset --hard HEAD~1) and observe the changes.

Screenshot as Q4_hard_reset.png

```
user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$ git reset --hard HEAD~1
HEAD is now at f585eac first commit

user@DESKTOP-6S448KE MINGW64 ~/lab3 (main)
$
```