WEEK 8

Mandatory Hands-On

**GIT and Version Control**

**Merge conflicts and resolving it**

**1. Detect the Conflict**

When you run:

git merge branch\_name

Git may stop and show:

CONFLICT (content): Merge conflict in filename.txt

Automatic merge failed; fix conflicts and then commit the result.

**2. Check Which Files Have Conflicts**

git status

You’ll see:

Unmerged paths:

(use "git add <file>..." to mark resolution)

both modified: filename.txt

**3. Open the Conflicted File**

Inside the file, Git adds **conflict markers** like this:

<<<<<<< HEAD

This is your branch's content

=======

This is the other branch's content

>>>>>>> branch\_name

* **<<<<<<< HEAD** → Your current branch’s changes.
* **=======** → Separator.
* **>>>>>>> branch\_name** → Changes from the branch you are merging.

**4. Decide How to Resolve**

You have three choices:

1. Keep your changes only
2. Keep the other branch’s changes only
3. Combine both changes manually

Edit the file to keep only what you want, and remove the conflict markers.

Example final resolved file:

This is the combined content from both versions.

**5. Mark as Resolved**

After editing, tell Git that you’ve resolved the conflict:

git add filename.txt

**6. Complete the Merge**

git commit

Git will use a default merge commit message (you can edit it if needed).

**7. Verify**

git log --graph --oneline

This confirms that the merge commit was created.

If conflicts are too complex, you can use:

git merge --abort

to cancel the merge and try again later.

**Sample Output:**





