Exercise 6B Report: Rebase and Clean Project History

Introduction: In this exercise, we utilized the repository from Exercise 6A to practice changing the message of the last commit and cleaning the project history using Git's interactive rebase feature. The objective was to modify the commit message to "I can change the last commit message" and then use Git rebase in interactive mode to squash commits together, cleaning the project history.

What I Did:

1. **Changed the Last Commit Message:** Initially, the last commit message was altered to "I can change the last commit message" using the git commit --amend command. This allowed me to adjust the commit message and ensure that it aligned with the exercise's requirements.
2. **Used Git Rebase in Interactive Mode:** To clean up the commit history, I used the git rebase -i HEAD~3 command to rebase the last three commits. This allowed me to interactively modify the commit history, change commit order, and squash multiple commits into a single one.

The rebase process included:

   - **Pick**: The command pick was used to keep the commit "Remove unwanted files" intact.
   - **Squash**: The squash command was used to combine commits "Restore 'new_file' after git rm --cached" and "Initial commit" into a single, cohesive commit.

3. **Fixed Conflicts:** During the rebase, there were some conflicts that had to be resolved manually. After resolving the conflicts, I continued the rebase with the git rebase --continue command to finalize the process.
4. **Final Commit History Cleanup:** After completing the rebase, I checked the commit history using git log --oneline to confirm that the commits were successfully squashed and the history was clean.

Challenges Faced:

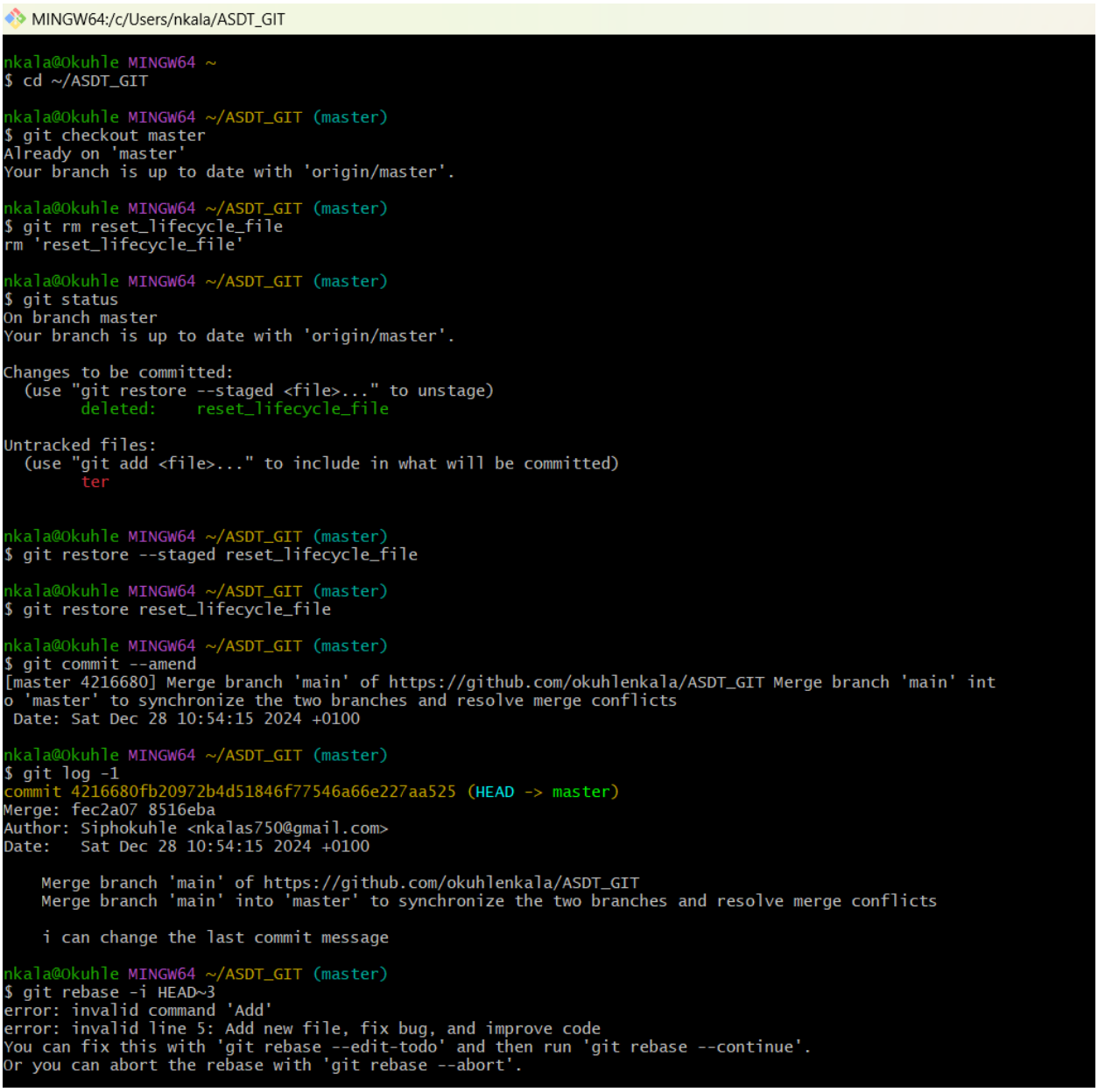
1. **Interactive Rebase Conflicts:** During the rebase, there were conflicts that needed to be addressed manually. Although conflicts are a common occurrence in rebase operations, resolving them took a bit of time and care to ensure that the commit history remained intact.
2. **Untracked Files:** While working through the rebase process, an untracked file (ter) was detected. This required additional steps to either remove or commit the file to ensure it was handled properly in the final commit history.

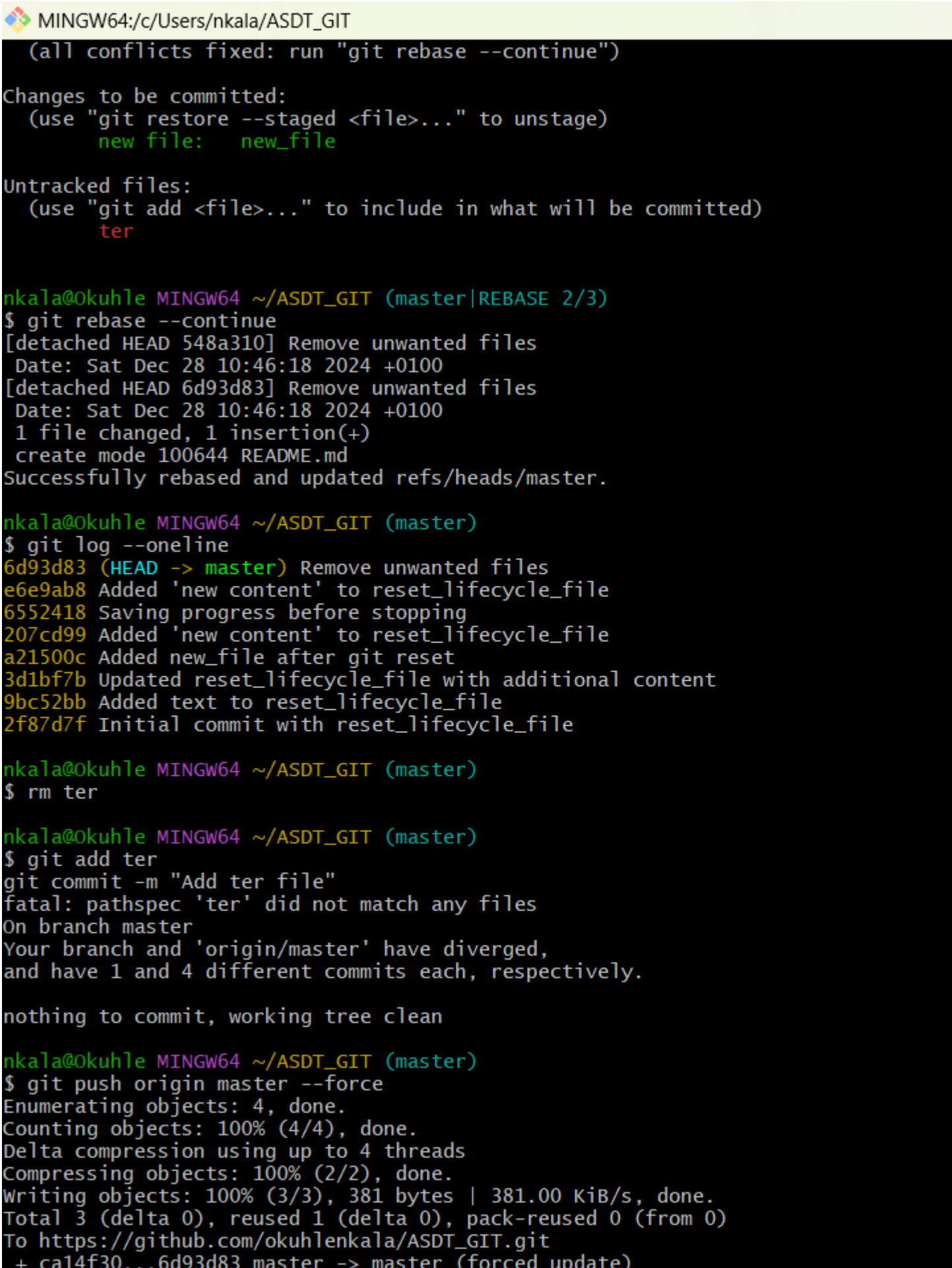
3. **Rebasing with Errors:** I encountered an issue where an invalid command (such as Add) was used in the rebase, resulting in an error. This required me to edit the rebase todo list and manually resolve the problem before continuing with the rebase.

Conclusions:

1. **Learning Outcomes:** The exercise helped solidify the understanding of the git rebase command, particularly in interactive mode. This mode is useful for cleaning up commit histories, rewriting commit messages, and squashing multiple commits into a single one.
2. **Rebase Best Practices:** It's important to carefully review each step during an interactive rebase to avoid errors and conflicts. Additionally, keeping track of untracked files is essential to prevent leaving unnecessary files in the repository.
3. **Git Rebase as a Powerful Tool:** Git rebase provides a powerful way to clean up project history and make it more readable, especially when preparing a project for collaboration or public release. It helps to maintain a linear history and avoids unnecessary commits.

4.


MINGW64:/c/Users/nkala/ASDT_GIT
nkala@Okuhle MINGW64 ~
$ cd ~/ASDT_GIT
nkala@Okuhle MINGW64 ~/ASDT_GIT (master)
$ git checkout master
Already on 'master'
Your branch is up to date with 'origin/master'.
nkala@Okuhle MINGW64 ~/ASDT_GIT (master)
$ git rm reset_lifecycle_file
rm 'reset_lifecycle_file'
nkala@Okuhle MINGW64 ~/ASDT_GIT (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
deleted: reset_lifecycle_file
Untracked files:
(use "git add <file>..." to include in what will be committed)
ter
nkala@Okuhle MINGW64 ~/ASDT_GIT (master)
$ git restore --staged reset_lifecycle_file
nkala@Okuhle MINGW64 ~/ASDT_GIT (master)
$ git restore reset_lifecycle_file
nkala@Okuhle MINGW64 ~/ASDT_GIT (master)
$ git commit --amend
[master 4216680] Merge branch 'main' of https://github.com/okuhlenkala/ASDT_GIT Merge branch 'main' int
o 'master' to synchronize the two branches and resolve merge conflicts
Date: Sat Dec 28 10:54:15 2024 +0100
nkala@Okuhle MINGW64 ~/ASDT_GIT (master)
$ git log -1
commit 4216680fb20972b4d51846f77546a66e227aa525 (HEAD -> master)
Merge: fec2a07 8516eba
Author: Siphokuhle <nkalas750@gmail.com>
Date: Sat Dec 28 10:54:15 2024 +0100
Merge branch 'main' of https://github.com/okuhlenkala/ASDT_GIT
Merge branch 'main' into 'master' to synchronize the two branches and resolve merge conflicts
i can change the last commit message
nkala@Okuhle MINGW64 ~/ASDT_GIT (master)
$ git rebase -i HEAD~3
error: invalid command 'Add'
error: invalid line 5: Add new file, fix bug, and improve code
You can fix this with 'git rebase --edit-todo' and then run 'git rebase --continue'.
Or you can abort the rebase with 'git rebase --abort'.


MINGW64:/c/Users/nkala/ASDT_GIT
(all conflicts fixed: run "git rebase --continue")
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
new file: new_file
Untracked files:
(use "git add <file>..." to include in what will be committed)
ter
nkala@Okuhle MINGW64 ~/ASDT_GIT (master|REBASE 2/3)
$ git rebase --continue
[detached HEAD 548a310] Remove unwanted files
Date: Sat Dec 28 10:46:18 2024 +0100
[detached HEAD 6d93d83] Remove unwanted files
Date: Sat Dec 28 10:46:18 2024 +0100
1 file changed, 1 insertion(+)
create mode 100644 README.md
Successfully rebased and updated refs/heads/master.
nkala@Okuhle MINGW64 ~/ASDT_GIT (master)
$ git log --oneline
6d93d83 (HEAD -> master) Remove unwanted files
e6e9ab8 Added 'new content' to reset_lifecycle_file
6552418 Saving progress before stopping
207cd99 Added 'new content' to reset_lifecycle_file
a21500c Added new_file after git reset
3d1bf7b Updated reset_lifecycle_file with additional content
9bc52bb Added text to reset_lifecycle_file
2f87d7f Initial commit with reset_lifecycle_file
nkala@Okuhle MINGW64 ~/ASDT_GIT (master)
$ rm ter
nkala@Okuhle MINGW64 ~/ASDT_GIT (master)
$ git add ter
git commit -m "Add ter file"
fatal: pathspec 'ter' did not match any files
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 4 different commits each, respectively.
nothing to commit, working tree clean
nkala@Okuhle MINGW64 ~/ASDT_GIT (master)
$ git push origin master --force
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 381 bytes | 381.00 KiB/s, done.
Total 3 (delta 0), reused 1 (delta 0), pack-reused 0 (from 0)
To https://github.com/okuhlenkala/ASDT_GIT.git
+ ca14f30...6d93d83 master -> master (forced update)