Lab Exercise 3- Working with Git Rebase

Lab Exercise: Git Rebase

This exercise demonstrates the use of git rebase in a scenario where no conflicts occur.

Objective

1. Learn how to rebase branches when there are no conflicting changes.

2. Understand the clean, linear history created by git rebase.

Prerequisites

1. Install Git on your system.

2. Initialize a Git repository:

```
git init git-rebase-lab

cd git-rebase-lab
```

Step-by-Step Instructions

1. Set Up the Repository

1. Create the main branch and make the initial commit:

```
echo "Line 1 from main branch" > file.txt

git add file.txt
```

git commit -m "Initial commit: Add Line 1 from main branch"

2. Create a new branch feature-branch:

```
git checkout -b feature-branch
```

3. Add a new line to file.txt in feature-branch:

```
echo "Line 2 from feature branch" >> file.txt

git add file.txt

git commit -m "Add Line 2 from feature branch"
```

4. Switch back to the main branch and add another line:

```
git checkout main

echo "Line 3 from main branch" >> file.txt

git add file.txt

git commit -m "Add Line 3 from main branch"
```

2. Rebase feature-branch onto main

1. Switch to feature-branch:

```
git checkout feature-branch
```

2. Rebase feature-branch onto main:

```
git rebase main
```

3. Git will replay the commit from feature-branch onto the main branch. Since there are no conflicts, the rebase completes automatically.

---

3. Verify the Rebase

1. View the commit history:

```
git log --oneline

git log --oneline --graph
```

Example:

2. Check the contents of file.txt:

```
cat file.txt
```

Output:

```
shagu@Shagun MINGW64 ~/git-reset-lab/git-rebase-lab (feature-branch)
$ cat file.txt
Line 1 from main branch
line 2 from feature branch
line 3 from main branch
```