Assignment 2

1] Initialize a new git repository in a directory of your choice. Add a simple text file to the repository and make the first commit.

Here's a step-by-step to initialize a new Git repository, add a simple text file, and make the first commit:

1. Open the terminal or command prompt.
2. Navigate to the directory where you want to create your Git repository using the cd command. For example:

------- cd path/to/your/directory

Initialize a new Git repository using the git init command:

--------git init

Create a simple text file. You can use any text editor to create the file. For example, you can create a file named example.txt with the following content:

-------This is a simple text file.

Add the text file to the Git staging area using the git add command:

--------git add example.txt

Commit the changes to the repository using the git commit command:

-------git commit -m "Initial commit: Add example.txt"

In this command, -m is used to add a commit message. Make sure to provide a descriptive message that explains the changes you made in this commit.

Now, you've initialized a new Git repository, added a simple text file, and made the first commit. You can continue making changes to your files, staging them with git add, and committing them with git commit as needed.

2] Branch Creation and switching

Create a new branch named ‘feature’ and switch to it. Make changes in the ‘feature’ branch and commit them.

Here are the steps to create a new branch named 'feature', switch to it, make changes, and commit them:

Create a new branch named 'feature' using the git branch command:

----git branch feature

Switch to the 'feature' branch using the git checkout command:

----git checkout feature

We can use the git switch command if you have a Git version that supports it:

----git switch feature

Make changes to your files in the 'feature' branch.

Add the modified files to the staging area using the git add command:

----git add .

This command adds all modified files to the staging area. If you want to add specific files, replace ‘.’ with the file names.

Commit the changes to the 'feature' branch using the ‘ git commit’ command:

----git commit -m "Made changes in the feature branch"

3] Create a ‘hotfix’ branch to fix an issue in the main code. Merge the ‘hotfix’ branch into ‘main’ ensuring that the issue is resolved.

The steps to create a 'hotfix' branch to fix an issue in the main code and then merge the 'hotfix' branch into 'main' to ensure that the issue is resolved:

1. Create a new branch named 'hotfix' based on 'main' using the ‘git checkout –b’ command:

   ---git checkout -b hotfix main

   This command creates a new branch named 'hotfix' and switches to it. The 'hotfix' branch will be based on the 'main' branch.

2. Add the modified files to the staging area:

   ---git add .

3. Commit the changes to the 'hotfix' branch:

   ---git commit -m "Fixed issue in main code"

4. Switch back to the 'main' branch:

   ---git checkout main

5. Merge the changes from the 'hotfix' branch into 'main':

   ---git merge hotfix

This command will merge the changes from the 'hotfix' branch into the 'main' branch.

6. Once the merge is successful and any conflicts are resolved, commit the merge:

   ---git commit -m "Merged hotfix into main"

   This commit message indicates that the 'hotfix' branch has been successfully merged into 'main'.

   Now, the issue in the main code has been fixed by the changes made in the 'hotfix' branch, and the 'hotfix' branch has been merged into 'main', ensuring that the issue is resolved in the main codebase.