Day-07

Assignment 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.

→Solution :

Open a terminal and navigate to the directory where you want to create the WiproTraining directory.

Step-1. Create the WiproTraining directory using the mkdir command:

mkdir WiproTraining

Step-2. Navigate to the WiproTraining directory using the cd command:

cd wiprotraining

Step-3. Initialize a new Git repository using the git init command:

git init

Step-4. Create a new file named BioData.txt using a text editor of your choice (e.g., nano, vim, emacs, etc.) and add the text "that shit" to it.

Add the BioData.txt file to the Git repository using the git add command:

git add BioData.txt

Step-5. Commit the changes to the Git repository using the git commit command:

git commit -m "Initial commit with BioData.txt"

Step-6. You can verify that everything worked correctly by checking the status of the repository:
git status
This command will show you the current status of the repository.

:: After running these commands, you should have a new Git repository in the WiproTraining directory with a single commit that includes the BioData.txt file with the text "that shit" in it.

Assignment 2: Branch Creation and Switching

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

→Solution :

Step-1. This will create a new branch called 'feature' and switch to it. Now you can make changes in this branch.

git checkout -b feature

Step-2. you want to modify the BioData.txt file in the 'feature' branch, you can use a text editor to modify the file and then stage and commit the changes using the following commands:

git add BioData.txt

git commit -m "Updated BioData.txt in feature branch"

: This will stage the changes in BioData.txt and commit them to the 'feature' branch.

Step-3. To switch back to the 'master' branch, you can use the following command:

git checkout master

:: This will switch your working directory to the 'master' branch.

Assignment 3: Feature Branches and Hotfixes

Create a 'hotfix' branch to fix an issue in the main code. Merge the 'hotfix' branch into 'main' ensuring that the issue is resolve .

→Solution :

Step-1. First, switch to the 'main' branch:

git checkout main

Step-2. Create a new 'hotfix' branch:

git checkout -b hotfix

Step-3&4. Make the necessary changes to fix the issue in the 'hotfix' branch. For example, you can modify the BioData.txt file to fix a typo.

Stage and commit the changes in the 'hotfix' branch:

git add BioData.txt

git commit -m "Fixed typo in BioData.txt"

Step-5. Switch back to the 'main' branch:

git checkout main

Step-6. Merge the 'hotfix' branch into 'main':

git merge hotfix

This command merges the changes from the 'hotfix' branch into the 'main' branch. If there are no conflicts, Git will automatically merge the branches. If there are conflicts, you'll need to resolve them manually.

Step-7. Verify the fix:

Once the merge is successful, verify that the issue is resolved by testing your code.

:: I 've successfully created a 'hotfix' branch to fix an issue in the main code and merged it into 'main' to ensure that the issue is resolved.