**Assignment 1: Initialize a new Git repository in a directory of your choice. Add a simple In40 text file to the repository and make the first commit.**

1. Choose a Directory:
    - Choose Project folder
2. Open Terminal/Command Prompt:
    - Navigate to your chosen directory using the **cd** command in your terminal.
3. Initialize Git Repository:
    - Use the "**git init"**command to create a new Git repository in the current directory.
4. Create and Add In40 Text File:
    - Create a new text file named "in40.txt" using text editor.
    - Then use the git add In40.txt command to add the file to the Git staging area. This tells Git that you want to include this file in the next commit.
5. Commit the Changes:
    - Use the command git commit -m "message" to create a snapshot of the current project state. The -m flag specifies a message describing the commit.

This will initialise a new Git repository, add an In40 text file, and make the first commit.

---

**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.**

**Ans:**

1. Verify Current Branch (Optional):
    - Use the git branch command to see which branch you're currently on. Ideally, this should be the master branch created in Assignment 1.
    - bash
2. Create and Switch to 'feature' Branch:
    - Use the following command to create a new branch named 'feature' and switch to it simultaneously:
        - Explanation:
            - git checkout: this command is used to switch branches.

- -b flag: tells Git to create a new branch named 'feature' and switch to it.

3. Make Changes in 'feature' Branch:
    - Now that you're in the 'feature' branch, edit your "in40.txt" file or create new files specific to your feature development.
4. Commit Changes in 'feature' Branch:

   First stage all the changes and then you have to commit

   git add . (staging all changes)

   git commit -m “message”

This will create a new branch named 'feature', switch to it, make changes, and commit them to the 'feature' branch. This allows you to develop your feature independently without affecting the main project code (usually in the 'master' branch).

---

**Assignment 3: Feature Branches and Hotfixes**

1. Identify the Issue and Branch from 'master':
    - Use the following command to create a new branch named 'hotfix' from the current 'master' branch: **git checkout -b hotfix**
        - This creates a new 'hotfix' branch that diverges from the exact state of the 'master' branch.
2. Fix the Issue in 'hotfix' Branch:
    - Switch to the 'hotfix' branch using: git checkout hotfix
    - Edit the "in40.txt" to fix the issue.
3. Commit the Fix in 'hotfix' Branch:
    - Once you've fixed the bug, stage your changes using **git add in40.txt**
    - Commit your fix with a clear message using : git commit -m “fixed issue in in40.txt”
4. Merge 'hotfix' Branch into 'master':
    - Switch back to the 'master' branch using:
    - Merge the 'hotfix' branch into 'master' to integrate the fix. Use the following command: git merge hotfix
5. Resolve Merge Conflicts (if any):

    - If there are merge conflicts, Git will signal this and highlight the conflicting parts in the files. Manually edit the files to resolve these conflicts and then add and commit the resolved files.
6. Verify the Fix (Optional):
    - Switch back to the 'master' branch and build/test your code to ensure the hotfix resolved the issue.
7. Push Changes (Optional):
    - If you're working with a remote Git repository, you can push your changes to the remote server after a successful merge:
    - git push origin master