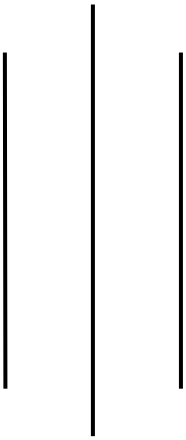
PURBANCHAL UNIVERSITY

KHWOPA ENGINEERING COLLEGE

LIBALI-08, BHAKTAPUR



LAB REPORT ON .NET

LAB NO. 01

SUBMITTED BY:

Name: Saragam Adhikari

Roll No. 770337

Group: B

SUBMITTED TO:

Department of Computer Engineering

Submission Date: 2081/12/10

Theory:

1. **GitHub:**
   GitHub is a cloud-based platform that hosts Git repositories, allowing developers to store, manage, and collaborate on code. It provides features like pull requests, issue tracking, and project management tools to streamline development. With GitHub, teams can work together on code, review changes, and merge updates efficiently. It also offers integration with CI/CD pipelines, making automated testing and deployment easier. Public repositories are free, while private ones require a subscription. Overall, GitHub enhances Git's functionality by providing a user-friendly interface and collaboration tools.

2. **Git:**
   Git is a distributed version control system that helps track changes in code, making collaboration easier. It allows developers to work on different features using branches without affecting the main code. Git stores project history, enabling users to revert to previous versions if needed. With Git, multiple people can work on the same project simultaneously without conflicts. It efficiently manages code updates, ensuring smooth integration of changes. Common commands like **git init, git add, git commit,** and **git push** help in managing code effectively.

General Git and GitHub Commands:

Git Configuration

git config --global user.name "Your Name"

Assigns a global username for all Git commits made on the system.

git config --global user.email "your_email@example.com"

Sets the global email address for Git commits on the system.

Initializing

git init

Initializes a new Git repository in the current directory, creating a `.git` folder to track changes.

Staging and Commits

git add .

It stages all changes and new files for commit.

git commit -m "Commit message as u like"

Saves the staged changes with a descriptive commit message.

Branching and Merging:-

git branch

Lists all the branches in the repository and highlights the current active branch..

git branch <branch_name>

Creates a new branch with the specified name, allowing you to work on separate features or tasks without affecting the main code.

git checkout <branch_name> / Git switch <branch_name>

Switches to the specified branch

git merge <branch_name>

Merges the specified branch into the current branch, combining changes from both branches.

Pushing and Pulling

git push -u origin <branch_name>

Pushes the specified branch to the remote repository and sets the **upstream(-u),** so future pushes can be done with just **`git push`.**

git pull origin <branch_name>

Fetches and merge the latest changes from the remote repository into the current local branch.

Status and Logs

git status

Show the current state of the files in the working directory (modified, staged or untracked).

git log

Displays the commit history of the repository showing details like commit IDs, authors, dates, and messages.

GitHub Specific

git remote add origin <repo_url>

Links the local repository to a remote repository on GitHub, enabling push and pull operations.

Lab Works:-

Setting the global username and email of the GitHub.

```
sarag@Saragam MINGW64 ~ (master)
$ git config user.name saragamadhikari

sarag@Saragam MINGW64 ~ (master)
$ git config user.email saragam.adhikari11@gmail.com
```

Create a folder and add files of your choice inside it. This will help track changes using Git for version control.

```
PS D:\Lab_Git\firstlab> git init
Initialized empty Git repository in D:/Lab_Git/firstlab/.git/
PS D:\Lab_Git\firstlab> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.txt
        text.py

nothing added to commit but untracked files present (use "git add" to track)
PS D:\Lab_Git\firstlab>
```

When new files are created, they are initially untracked. To track them, first initialize the directory with Git, then add the files to the staging area, and finally commit them to the repository.

```
PS D:\Lab_Git\firstlab> git add .
PS D:\Lab_Git\firstlab> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   test.txt
        new file:   text.py
```

Now, commit the files to store them in the local repository, ensuring all tracked changes are saved.

```
PS D:\Lab_Git\firstlab> git commit -m "New files "
[master (root-commit) 9e6769e] New files
 2 files changed, 2 insertions(+)
 create mode 100644 test.txt
 create mode 100644 text.py
```

Modify the file to observe changes in its status, allowing you to track updates using Git.

```
PS D:\Lab_Git\firstlab> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   text.py

no changes added to commit (use "git add" and/or "git commit -a")
```

After modifying text.py, add and commit the changes. Then, create a GitHub repository, copy its URL, link it to your local repository, and push the changes.

```
PS D:\Lab_Git\firstlab> git remote add origin https://github.com/saragamadhikari/dotnet.git
```

Now push the files in the repository created.

```
PS D:\Lab_Git\firstlab> git push origin main
>>
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (10/10), 889 bytes | 889.00 KiB/s, done.
Total 10 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/saragamadhikari/dotnet.git
   750eec0..3814443  main -> main
```

Now, create branches to work on different versions of the project. This allows you to make changes or experiment with new features without altering the main codebase. Each branch can hold a different version, keeping your main project intact while you test or develop new ideas.

```
PS D:\Lab_Git\firstlab> git branch feature
PS D:\Lab_Git\firstlab> git branch
  feature
* main
```

Switch to the new branch to modify the file without affecting the main codebase.

```
PS D:\Lab_Git\firstlab> git branch
* feature
  main
PS D:\Lab_Git\firstlab> git add .
PS D:\Lab_Git\firstlab> git status
On branch feature
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   calculation.py

PS D:\Lab_Git\firstlab> git commit -m "new file as calculations.py"
[feature 1255420] new file as calculations.py
 1 file changed, 1 insertion(+)
 create mode 100644 calculation.py
```

To switch branches, use the command ***git switch main.*** To make the branch visible to others, push it to GitHub.

```
PS D:\Lab_Git\firstlab> git push -u origin feature
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 364 bytes | 364.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:      https://github.com/saragamadhikari/dotnet/pull/new/feature
remote:
To https://github.com/saragamadhikari/dotnet.git
 * [new branch]      feature -> feature
branch 'feature' set up to track 'origin/feature'.
```

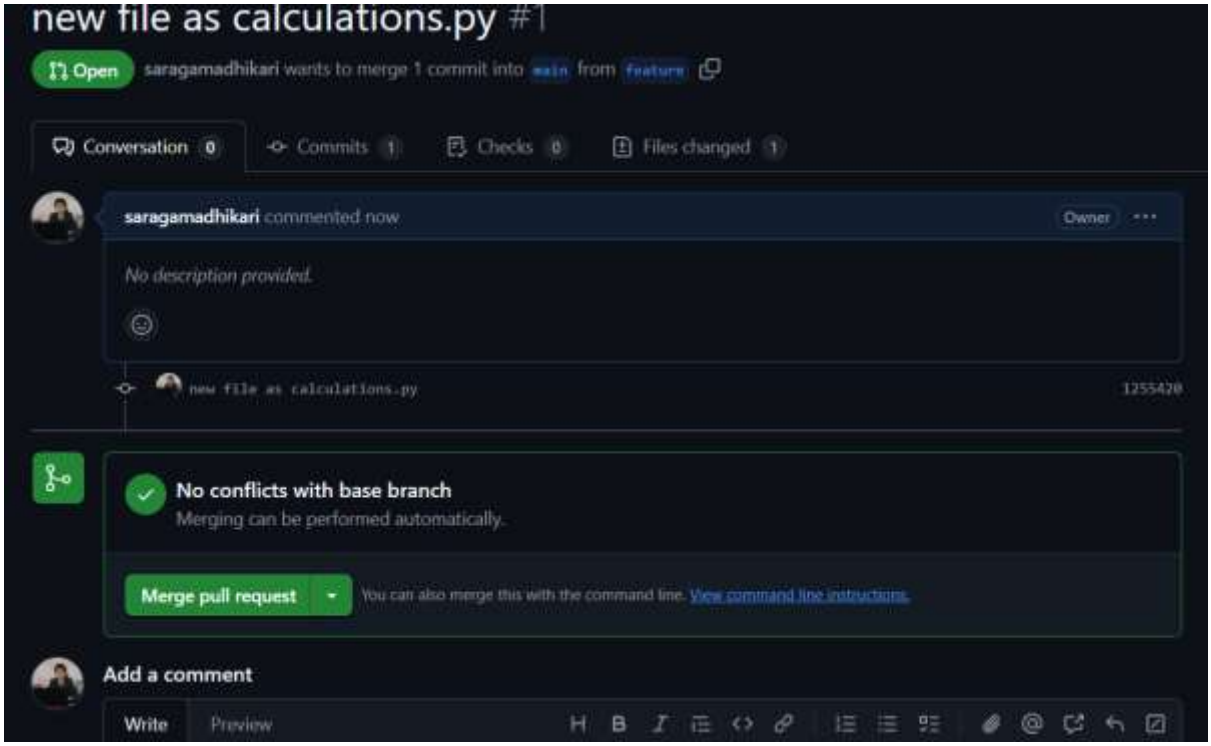
Before merging, check for differences between the main branch and the new branch to avoid conflicts. Once you're sure there are no issues, merge the new branch into the main branch, ensuring that the changes or new features from the new branch are integrated into the main codebase.

```
PS D:\Lab_Git\firstlab> git merge feature
Already up to date.
PS D:\Lab_Git\firstlab> git diff main
diff --git a/calcu.py b/calcu.py
new file mode 100644
index 0000000..68067c8
--- /dev/null
+++ b/calcu.py
@@ -0,0 +1 @@
+print("hello:")
\ No newline at end of file
diff --git a/calculation.py b/calculation.py
new file mode 100644
index 0000000..107ccaa
--- /dev/null
+++ b/calculation.py
@@ -0,0 +1 @@
+a = 100
\ No newline at end of file
```

Use the "***git log"*** command to view **past commits and their details**.

```
commit 381444343cfba501d462d61c3122283a22011496
Author: saragamadhikari <saragam.adhikari11@gmail.com>
Date:   Fri Mar 21 17:13:36 2025 +0545

    updates

commit 9a7b506692d91fb6b4e8faa00e7ffb2ba3eb2f54
commit 5f1d7b0f467386f5f06b4007f59e4884d56c0d27 (HEAD -> feature, main)
Author: saragamadhikari <saragam.adhikari11@gmail.com>
Date:   Fri Mar 21 17:48:10 2025 +0545

    new file as calcu

commit 1255420f4b53ec6c7dfb1f274d9981b051558419
Author: saragamadhikari <saragam.adhikari11@gmail.com>
Date:   Fri Mar 21 17:37:03 2025 +0545

    new file as calculations.py
```

To merge a branch in GitHub (Web), go to the repository, click "Pull requests," create a new pull request, select the branch to merge, and then click "Merge pull request."



Conclusion:

In this lab, we explore the basics of Git and GitHub, including initializing a repository, creating branches, merging changes, making commits, and pushing updates.