PURBANCHAL UNIVERSITY

KHWOPA ENGINEERING COLLEGE

LIBALI-08, BHAKTAPUR

LAB REPORT ON .NET

LAB NO. 01

SUBMITTED BY:

Name: Rajesh Hamal

Roll No. : 770329

Group: B

SUBMITTED TO:

Department of Computer Engineering

Submission: 2081/12/09

Theory:

1. **Git:**

   Git is a distributed version control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

   Git is a **distributed version control system (DVCS)** designed to help multiple developers collaborate on a project without conflicts or data loss. It enables efficient tracking of changes, version management, and merging of contributions from different developers.

   Key Purposes of Git as a DVCS:

a) Distributed System:

Unlike centralized version control systems (e.g., SVN), Git does not rely on a single central repository. Each developer has a complete copy of the repository, including the full history, allowing them to work offline and independently.

b) Collaboration Without Overwrites:

Developers work on separate branches, preventing direct conflicts. Changes are merged strategically using Git's merging and rebasing features.

c) Version Control & History Tracking:

Every change is recorded with a commit message, making it easy to track who changed what and when. Previous versions of the project can be restored if needed.

d) Branching & Merging:

Developers can create isolated branches for new features or bug fixes. Once tested, changes can be merged into the main branch without disrupting others' work.

e) Conflict Resolution:

If multiple developers edit the same file, Git detects conflicts and allows manual resolution before finalizing changes.

f) Collaboration via Remote Repositories:

Platforms like GitHub, GitLab, and Bitbucket enable developers to push and pull changes to a shared repository. Teams can work asynchronously and from different locations.

Git's primary features include:

- Version Control: Git keeps a history of changes made to the codebase, allowing developers to revert to previous versions if needed.
- Branching and Merging: Developers can create branches to work on new features or fixes independently and merge them back into the main branch once they are ready.
- Distributed Workflow: Git allows each developer to have a complete copy of the repository, enabling offline work and reducing dependency on a central server.

2. GitHub

GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. GitHub supports features like branching, pull requests, issue tracking, and CI/CD integration. It is widely used for open-source and private projects, enabling seamless teamwork. GitHub also provides cloud-based hosting, making it accessible from anywhere.

Key Features of GitHub for Collaboration

a) Pull Requests (PRs) – Code Review & Merging:
Allows developers to propose changes, review them, and merge them into the main project.

b) Issues – Bug Tracking & Feature Requests:
Acts as a built-in ticketing system for reporting bugs, requesting features, and discussing improvements.

c) Project Management Tools (GitHub Projects & Discussions):
Helps teams organize tasks and workflows directly within GitHub.

d) Actions & CI/CD – Automating Workflows
Automates testing, deployment, and other development tasks.

e) Forking & Open Source Contributions
Encourages open-source collaboration by allowing anyone to copy a repository, modify it, and contribute back.

It builds on Git's capabilities and offers:

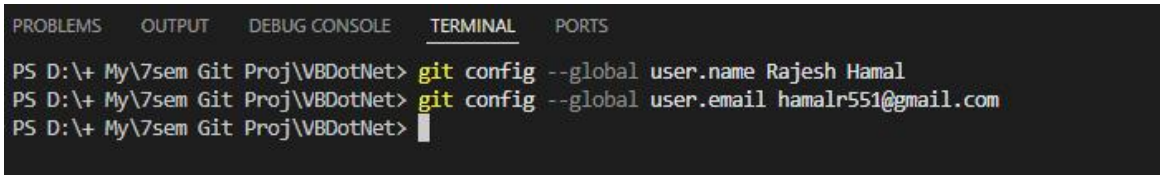
- Repository Hosting: GitHub stores your code in repositories, making it accessible from anywhere.
- Collaboration Tools: GitHub allows multiple developers to work on the same project, review each other's code, and suggest improvements through pull requests.
- Project Management: GitHub provides tools for tracking issues, managing tasks, and organizing projects.

General Git and GitHub Commands:

Commands	Functions
git init	Initializes a new Git repository in the current directory.
git add .	Stages all changed and new files for the next commit.
git commit -m “message”	Commits staged changes with a descriptive message.
git status	Shows the current status of the working directory and staging area.
git log	Displays a history of commits in the repository.
git add file	Stages a specific file for the next commit.
git pull origin branch_name	Fetches and merges updates from a remote repository.
git push origin branch_name	Pushes a specific branch to the remote repository.
git remote -v	Lists all remote repositories linked to the project.
git merge branch_name	Merges the specified branch into the current branch.
git branch	Lists all local branches in the repository.
git branch branch_name	Creates a new branch.
git checkout branch_name	Switches to a different branch.
git checkout -b branch_name	Creates and switches to a new branch.
git clone repo_url	Copies (clones) a remote repository to your local machine.
git remote add origin repo_link	Add the local repository with the github repository specified by the url link.

Lab Works

1. First set the global username and email of the GitHub.



2. Create a folder and files inside it so that we can identify the changes inside the file using the version control (Git).

On creating the new files, initially the files are in the untracked stage so sent the untracked files to the staging stage. To do so first initialize the directory and staged the files

```
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1> git status
On branch newbranch
Your branch is up to date with 'origin/newbranch'.

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

nothing added to commit but untracked files present (use "git add" to track)
```

As we can see a new file is created but it is not saved in the local repository so we use add command to do so.

```
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1> git add .
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1> git status
On branch newbranch
Your branch is up to date with 'origin/newbranch'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   newds.py

PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1>
```

3. Now commit the files such that the files are stored in the local repository.

```
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1> git commit -m "Adding new file"
[develop 0d0c735] Adding new file
 3 files changed, 50 insertions(+)
 create mode 160000 khec
 create mode 100644 new.py
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1>
```

4. To connect the local and github repository we use follwing command

```
PS D:\8th\.net\test> git remote add origin https://github.com/Surag-Basukala/.net-reports.git
```

5. Now to create and change the current branch of the repository

```
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1> git branch
* develop
  master
  new
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1> git branch just_created
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1> git branch
* develop
  just_created
  master
  new
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1> git checkout just_created
Switched to branch 'just_created'
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1>
```

6. Now to save the commited files in the github repository through specific branch

```
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1> git push origin develop
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 618 bytes | 618.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/rajesh6007/Vs.Net-Lab.git
   9478824..0d0c735  develop -> develop
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1>
```

Here develop is the branch through which files are pushed

7. Similary to pull the files form github to local repository.

```
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1> git pull origin develop
From https://github.com/rajesh6007/Vs.Net-Lab
 * branch            develop    -> FETCH_HEAD
Already up to date.
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1>
```

8. To clone the public github repository in the local repository.

```
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1> git clone https://github.com/rajesh6007/Flask-app.git
Cloning into 'Flask-app'...
remote: Enumerating objects: 324, done.
remote: Counting objects: 100% (324/324), done.
remote: Compressing objects: 100% (215/215), done.
remote: Total 324 (delta 117), reused 306 (delta 100), pack-reused 0 (from 0)
Receiving objects: 100% (324/324), 10.78 MiB | 12.64 MiB/s, done.
Resolving deltas: 100% (117/117), done.
PS D:\+ My\7sem Git Proj\VBDotNet\Git_lab1>
```

Conclusion:

In this lab, we learn about the basics of the Git and GitHub. We perform initialization, branching, merging, pushing and commit.