PREMIER UNIVERSITY
CHATTOGRAM

Department of Computer Science and Engineering

Lab Report

Course Title	: Software Engineering & Information System Design Lab.
Course Code	: CSE 306
Experiment No.	: 01
Name of Experiment	: Introduction to Git and Github.
Date of Experiment	: 11.11.2023
Date of Submission	: 18.11.2023

Submitted to:
MD. Tamim Hossain
Lecturer
Department of CSE
Premier University

Submitted by:

Name	: Md. Nurnabi Rana		
ID	: 210401020 **2290**		
Department	: CSE		
Batch	: 40th	Section	: D
Semester	: 5th		

Remarks:

MD. NURNABI RANA
2104010202290

Abstract : The following lab report aims to introduce GIT and GITHUB. The lab was conducted to address the need for effective version controll and collaborative software development. Including the succesful creation of a GIT repository, execution of commands and understanding file tracking mechanism. The lab explores the integration of GITHUB becomes a cruicial element in version controll process. The utilization of GITHUB enhances collaborative efforts by providing centralized platform for code hosting and the management of repositories.

Introduction :

GIT : Git is a distributed version controll system used for tracking changes in source code during software development. It allows multiple developers to collaborate on a project by managing and merging different versions of the database.

GITHUB : Github is a web-based platform that serves as a version controll system for tracking changes in code and facilitates collaboration among developers.

MD. NURNABI RANA
2104010202290

Materials :

1. Personal Computer
2. Internet Connection
3. GIT
4. GITHUB account
5. Text Editor or IDE.

Activity :

Activity 1 : Create a git repo.

1. Create a directory i want to set as my repository in a location :
   - $ mkdir myfirstRepo : This command creates a new directory (folder) named "myfirstRepo".
   - cd myFirstRepo : This command changes the current directory to "myfirstRepo".

2. Initialize the directory as a repository :

```
$ git init
$ git config --global init.defaultBranch main
$ git branch -m main
```

3. Use config to add my name and email :

```
$ git config --global user.name "2290"
$ git config --global user.email "xyz@gmail.com"
```

MD. NURNABI RANA
210 4010 202290

4. Create a new file in my repository:
   - Using a text editor such as notepad and save it to git repository.

☐ Activity 2: Create a python file (ex. first.py) that prints out "Hello World" and run it in the terminal

1. Create a python script in the directory:
   First.py
2. Inside the ~~code~~ file, write the given code:
   print("Hello World")
3. Add/Remove the file to/from the repository:
   - working directory: Untracked files (red); working directory - we can make changes in file and don't have to commit these changes.
   - straging area - Tracked files (green). Changes that are ready to be commited.
   - commit - Officialy saving the repository at a point in time.

   $ git status : To see staged & unstaged files.

a. $ git add "file name" : track file (move into straging area.

b. $ git rm --cached "file name" : untrack file (move back into working directory.

MD. NURNABI RANA
2104010202290

Activity 3 : Change first.txt and run it on the terminal, then add it, commit it, and do git status and git diff.

a) Add a file to staging :

- $ git status : Shows which state the file are in.
- $ git add first.py
- $ git status

b) Commits files in staging :

```
$ git commit -m "file added"
$ git log : to see what our commits look like.
$ git status
```

c) Change, save and exit out of the text file :

$ git diff : Show differences between commited file & current staged file.

d) Commit it again :

```
$ git commit -m "Edited current file"
$ git log
$ git status
```

MD. NURNABI RANA
2104010202290

GIT Command list :

• Git Configuration :

① git config --global user.name - Set the name that will be attached to user commits & tags.

② git config --global user.email "you@gmail.com" - Set the email address that will be attached to your commits & tags.

• Day-to-day work :

③ git status - Displays the status of working directory. Option inclued new, staged & modified files.

④ git add [file] - Add a file to the staging area.

⑤ git diff [file] - Show changes between working directory and staging area.

⑥ git checkout --[file] - Discard changes in working directory. The operation is unrecoverable

⑦ git commit - Create a new commit from changes added to the staging area. The commit must have a message.

⑧ git rm [file] - Remove file from working directory and staging area.

MD. NURNABI RANA
2104010202290

• **Storing Work:**

⑨ git stash - Put current changes in your working directory into stash for later use.

⑩ git stash pap - Apply stored stash content into working directory and clear trash.

⑪ git stash drop - Delete a specific stash from all previous stashes.

• **Git Branching Model:**

⑫ git branch [-a] - List all local branches in repository. Show all branches.

⑬ git branch [branch_name] - Create new branch, referencing the current HEAD.

⑭ git checkout [-b] [branch_name] - Switch working directory to the specified branch. Git will create branch if there is none

⑮ git merge [branch_name] - Join specified branch into current branch.

⑯ git branch -d [branch_name] - Remove selected branch.

MD. NURNABI RANA
2104010202290

• Inspect History :

(17) git log [-n count] – List commit history of current branch.

(18) git log ref.. – List commits that are represent on the current branch and not merged into ref.

(19) git log ..ref – List commit that are present on ref and not merged into current branch.

(20) git tag – List all tags.

(21) git tag -a [name] [commit sha] – Create a tag object named named "name" for current commit.

(22) git tag -d [name] – Remove a tag from local repository.

(23) git fetch [remote] – Fetch changes from the remote, but not update tracking branches.

(24) git pull [remote] – fetch changes from the remote and merge current branch with it's upsteam.

(25) git push -u [remote] [branch] – Push local branch to remote repository.

MD. NURNABI RANA
21040102022290

Discussion : While entering an incorrect Git command results in a clear error messages. These error was stemed from mistyped commands, unrecognized inputs, or permission issues. Git's feedback system helps us to identify and address errors efficiently.
To fix a wrong Git command, we correct the mistake based on the error message and following the instruction of course teacher (honourable).

Conclusion : As we Know Git is a crucial for versior controll, it enabling collaborative and efficient code development. With features like branching, traceability. Git ensures code integrity and automates deployment processes.
Git's future benefits lie in it's adapbility and ongoing evolution. As an essential tool for diverse projects, Git provides traceability, backup and recovery. Gits provides or plays a pivotal role in automating testing and deployment process, contributing to efficient and reliable Software development.