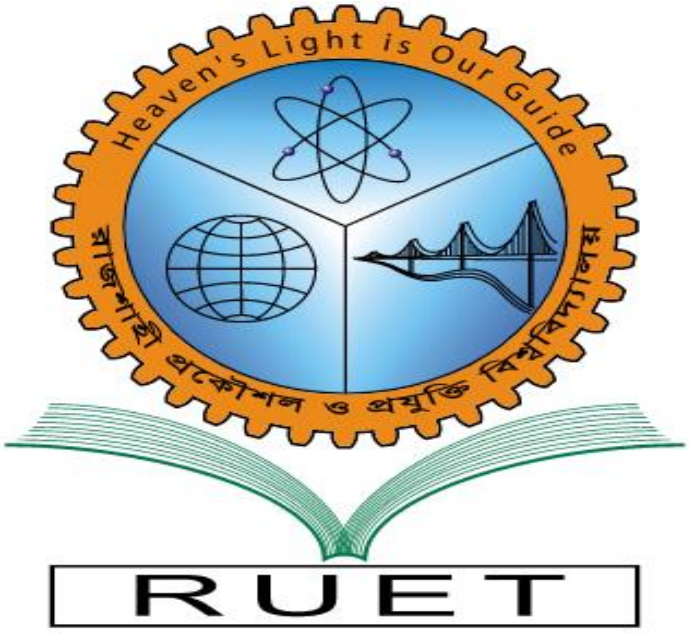
Heaven's Light Is Our Guide

Rajshahi University of Engineering and Technology, Rajshahi.



Course code: ECE 3117

Course title: Software Engineering and Information System Design

Lab Report 3

Submitted to:

Oishi Jyoti

Assistant Professor,

Department of ECE,RUET

Submitted by:

Md.Ashikul Islam

Roll:2010019

Task: Study of different git commands.

Objectives :

- To learn the importance and functionality of Git as a distributed version control system for managing codebases.
- To practice fundamental Git commands such as git init, git add, git commit, and git status for initializing repositories and tracking changes.
- To learn how to create branch using git branch.

Theory:

Git is a distributed version control system designed to manage and track changes in files, primarily source code, during software development. Unlike centralized version control systems, Git allows each user to have a complete copy of the repository, including the entire history of changes. This enables offline work and provides redundancy. Git makes it easy to create isolated branches for new features or bug fixes. These branches can later be merged into the main codebase, allowing parallel development. Thus Git enables multiple developers to work on the same project simultaneously, integrating changes from different contributors. Here are different Git commands:

1. **Git init Command:**

   Initializes a new Git repository in the current directory. Creates a .git folder to start tracking changes.

```
HP@Ashik MINGW64 ~/git-practice2
$ git init
Initialized empty Git repository in C:/Users/HP/git-practice2/.git/
```

2. **Git Config Command:**

   Configures user details like name and email globally or for a specific repository.

```
HP@Ashik MINGW64 ~/git-practice2 (master)
$ git config user.name "Aahikul Islam"

HP@Ashik MINGW64 ~/git-practice2 (master)
$ git config user.email "ashikul2486@gmail.com"
```

3. Git Add Command:

Stages changes (new or modified files) to prepare them for committing.

```
HP@Ashik MINGW64 ~/git-practice2 (master)
$ touch readme.txt

HP@Ashik MINGW64 ~/git-practice2 (master)
$ git add readme.txt
```

4.Git Commit Command:

Records staged changes into the repository's history. Requires a message describing the changes.

```
HP@Ashik MINGW64 ~/git-practice2 (master)
$ git commit -m "added text file"
[master (root-commit) e4d30fc] added text file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 readme.txt
```

5.Git Clone Command:

The git clone command is used to create a local copy of a remote repository.

```
HP@Ashik MINGW64 ~/git-practice2 (master)
$ git clone https://github.com/ashikul2486/Higithub.git
Cloning into 'Higithub'...
```

6.Git Status Command:

Displays the current state of the working directory and staging area. Shows untracked files, changes to be committed, and changes not yet staged.

```
HP@Ashik MINGW64 ~/git-practice2 (master)
$ git status
On branch master
```

7.Git Branch Command:

Lists all branches in the repository. Creates a new branch if a name is provided.

```
HP@Ashik MINGW64 ~/git-practice2 (master)
$ git branch asbranch
```

8.Git Push Command:

The git push command is used to upload local changes from a Git repository to a remote repository**.**

```
HP@Ashik MINGW64 ~/git-practice2 (master)
$ git push origin main
```

Conclusion: Git commands form the backbone of version control and collaboration in modern software development. Each command plays a unique role in managing a project's lifecycle, from initializing repositories to pushing changes to remote servers. By understanding and practicing these commands, developers can streamline their workflow, prevent data loss, and ensure code quality.