

Software Engineering Report

Subject Code: CS16106

Tool Name: Git



Purpose of Tool

Git is a free, open source and distributed version control system which can manage from small to very large projects efficiently by tracking changes, coordinating work among the collaborators and maintaining the integrity of the codebase.

Git can be accessed through the command line interface (CLI) as well as through graphical user interface (GUIs) available such as Github Desktop, Sourcetree etc.

Status of Tool

Git has evolved and matured to be easy to use and yet retain the initial qualities . It is amazingly fast and very efficient with large projects.

It has an incredible branch system for non-linear development such as Git Branching. Now the tool is in **working status**.

Specific Use of Git in SDLC Phases

Git plays a pivotal role in managing source code, effective collaboration, and ensuring software security throughout the Software Development Life Cycle (SDLC). Git finds its primary usage in the developmental stages of the Software Development Life Cycle (SDLC), and its application extends to various other stages, as listed below:

Planning: At this stage teams may structure projects preliminarily and define tasks.

Development: Git helps developers to create branches for features, bug fixes or experiments. Normally, each developer works on their own branch where changes can be merged back into the main codebase (usually master branch) when ready using git. Thus, everyone's work is separated and putting it together does not cause any problem.

Testing: Developers commit the changes into a shared repository. This is where Continuous Integration tools come in handy as they execute tests on these changes to make sure that the codebase remains stable and operational.

Deployment: Git makes it easy to tell exactly what updates are being deployed so that rollbacks can be done smoothly if needed.

Maintenance and Update: Through branching and merging capabilities of Git, all bug fixes, feature enhancements and changes are well handled.

Minimum System Requirements

Software Requirements	Hardware Requirements
OS: Windows 10 or newer ,Linux, macOS 10.13 or later 64-bit OS is required for running Git Desktop app.	There is no any minimum hardware requirement as such but some specifications are recommended
Browser: Chrome, Edge, Safari, Firefox	RAM: At least 1GB Internal Memory: Minimum of 50 MB of disk space,additional space for repository.

Inputs and Outputs

Git init

This command sets up a new git repository in a current directory, allowing users to track changes to the file.

```
MINGW64/c/Users/HP/Desktop/Software Engineering Project
HPDESKTOP-K4U004U MINGW64 ~/Desktop/Software Engineering Project
$ git init
Initialized empty Git repository in C:/Users/HP/Desktop/Software Engineering Project/.git/
HPDESKTOP-K4U004U MINGW64 ~/Desktop/Software Engineering Project (master)
$
```

Git add

“Git add” stages changes made to the file, preparing them to be added in the next commit.

```
HPDESKTOP-K4U004U MINGW64 ~/Desktop/Software Engineering Project (master)
$ git add .
HPDESKTOP-K4U004U MINGW64 ~/Desktop/Software Engineering Project (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .DS_Store
    new file:   dicee.html
    new file:   images/dice1.png
    new file:   images/dice2.png
    new file:   images/dice3.png
    new file:   images/dice4.png
    new file:   images/dice5.png
    new file:   images/dice6.png
    new file:   index.js
    new file:   styles.css
```

Git status

It displays what is happening in your repository, shows which files have been modified, and what changes are staged for the next commit.

```
HP@DESKTOP-K4U0U4U MINGW64 ~/Desktop/Software Engineering Project (master)
$ git status
On branch master
no commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .git_store
    dicee.html
    images/
    index.js
    styles.css

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

Git commit

It records the changes that have been made to the repository, creates a commit with a unique identifier and a message describing the committed changes.

```
HP@DESKTOP-K4U0U4U MINGW64 ~/Desktop/Software Engineering Project (master)
$ git commit -m "Initial Files Commit"
[master (root-commit) c8fcsb3] Initial Files Commit
10 files changed, 87 insertions(+)
create mode 100644 .git_store
create mode 100644 dicee.html
create mode 100644 images/dice1.png
create mode 100644 images/dice2.png
create mode 100644 images/dice3.png
create mode 100644 images/dice4.png
create mode 100644 images/dice5.png
create mode 100644 images/dice6.png
create mode 100644 index.js
create mode 100644 styles.css
```

Git branch

This command lets us see, create, delete, list, switch, rename branches with the help of various flags in a project. It allows users to organize branches effectively.

```
HP@DESKTOP-K4U0U4U MINGW64 ~/Desktop/Software Engineering Project (master)
$ git branch -M main
```

Git remote

This command helps users to manage remote repositories. It allows them to add, rename and remove remote repositories which are associated with local repositories.

Sub-commands: git remote add

git remote remove

git remote rename

```
HP@DESKTOP-K4U0U4U MINGW64 ~/Desktop/Software Engineering Project (main)
$ git remote add origin https://github.com/Vaibhav05kansal/Software-Engineering-Project.git
```

Git push

This command pushes or sends users local changes to the remote repository.

```
HP@DESKTOP-K4U0U4U MINGW64 ~/Desktop/Software Engineering Project (main)
$ git push -u origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (13/13), 14.08 KiB | 1.08 MiB/s, done.
Total 13 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Vaibhav05kansal/Software-Engineering-Project.git
   8009e1..705f9e: main -> main
branch 'main' set up to track 'origin/main'.
```

Git checkout

It allows users to switch between different branches in the repository or restore files to previous state. It helps users to navigate in the project history.

```
hp@LAPTOP-KKFLG05Q MINGW64 ~/Desktop/Software-Engineering-Project (main)
$ git checkout -b vikash
Switched to a new branch 'vikash'

hp@LAPTOP-KKFLG05Q MINGW64 ~/Desktop/Software-Engineering-Project (vikash)
$ git add .
```

Git clone

This command created a copy of a git repository from a remote source to your local system.

```
MINGW64/C:/Users/hp/Desktop
hp@LAPTOP-KKFLG05Q MINGW64 ~/Desktop
$ git clone https://github.com/VikashSingh0007/Software-Engineering-Project.git
Cloning into 'Software-Engineering-Project'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 16 (delta 0), reused 13 (delta 0), pack-reused 0
Receiving objects: 100% (16/16), 14.98 KiB | 95.00 KiB/s, done.
hp@LAPTOP-KKFLG05Q MINGW64 ~/Desktop
$ |
```

References

- Git Documentation (<https://git-scm.com/doc>)
- Github GUI (<https://github.com>)
- Github Repository (<https://github.com/Vaibhav05kansal/Software-Engineering-Project>)
- Plagiarism Check Tool (<https://www.duplichecker.com>)

Team Members

Sr. No.	Name	Registration No.	Signature
1	Vaibhav Kansal	20214225	
2	Vinayak Singh	20214306	
3	Vikash Singh	20214074	
4	Shashi Raj	20214263	
5	Shivam Kumar	20214008	

Plagiarism Check Report

