**Day 1**

**Git Version Control**

 A version control system is a software that tracks changes to a file or set of files over time so that you can recall specific versions later. It also allows us to work together with other programmers or developers.

**What is Git?**

Git is an open-source distributed version control system. It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

**Benefits of using Git**

 Major benefits of Git are:  
  
-Saves Time  
-Offline Working  
-Undo Mistakes  
-Track the Changes

**Instructions for configuration**

Git supports a command called git config that lets you get and set configuration variables that control all facets of how Git looks and operates. It is used to set Git configuration values on a global or local project level.  
  
Setting user.name and user.email are the necessary configuration options as your name and email will show up in your commit messages.

**Day 1 Reflections**

Git is a distributed version control system (VCS) designed to track changes in source code during software development. It allows multiple developers to collaborate on projects, keeping track of every modification made to the codebase over time.

**Day 2**

**Git Tools**

 Git Tools  
To explore the robust functionality of Git, we should learn about git tools. Git comes with some tools like Git Bash, Git GUI to provide the interface between machine and user.  
Git Package Tools  
Git provides powerful functionality tools such as commands, command line, Git GUI. Let's have a glimpse of some essential package tools.  
  
GitBash  
Git Bash is an application for the Windows environment. It is used as Git command line for windows.  
  
Git Bash also includes the full set of Git core commands like git clone, git commit, git checkout, git push, and more.  
-Git GUI  
Git GUI is a powerful alternative to Git BASH.  
$ git gui  
Git GUI Image 1  
-Gitk

**Git Terminology**

 -Git Reset  
-Git Ignore  
-Git Diff  
-Git Cheat Sheet  
-Git Flow  
-Git Squash  
-Git Rm  
-Git Fork

**Activity 1**

 Use Cases: Use git reset when you want to rewrite history, undo commits, or reapply commits differently.  
Use Cases: Use git merge for integrating changes from one branch into another while maintaining a linear history.  
Use git stash apply when you want to apply the stash but keep it available in the stash list for potential reuse.

**Day 3**

**Basic Git Command**

Git Config  
Git init  
Git clone  
Git add  
Git commit  
Git status  
Git push  
Git pull  
Git Branch  
Git Merge  
Git log  
Git remote

**Day 4**

**Inspect and Undo changes**

Git log  
The advantage of a version control system is that it records changes. These records allow us to retrieve the data like commits, figuring out bugs, updates. But, all of this history will be useless if we cannot navigate it. At this point, we need the git log command.

**Activity 1**

-git diff: Shows changes between the working directory and the staging area.  
-git status: Provides an overview of the current status of files.  
-git revert <commit>: Reverts the changes made in the specified commit.  
-git reset <commit>: Resets the branch to the specified commit, with different options like --soft, --mixed, or --hard affecting the staged area and working directory differently.  
-git rm <file>: Removes the specified file from the repository.

**Collaborating**

Collaborating in git git refers to sharing on code among several developers. We will look at code sharing is handled in Git.We will discuss major collaboration operations in Git. Our main focus will be on Git fetch, Git pull and Git Push.

**Activity 2**

Purpose: Merge integrates changes from one branch into another, creating a new commit that combines the changes of two branches.

**Activity 3**

These operations offer different ways to manage and integrate changes in a Git repository. Fetch and pull fetch changes from a remote repository, with pull immediately merging them. Squashing condenses commits, rebase restructures history, and merge combines branches, potentially requiring conflict resolution when changes overlap.

**Day 5**

**Activity 1**

The program uses a simple command-line interface to prompt the user for the type of conversion and the quantity to convert. Based on the user's choice, it performs the appropriate conversion and displays the result.