** GIT **
(Version Control System)

Doc contains -

- Commonly used Git commands
- Merge vs Rebase

Git very initial Setup -

➔ git config –global user.name “User Name”
➔ git config –global user.email “User Mail Id”

CREATE & Checkout to a branch

➔ git checkout -b branch_name [-b is used to create a branch if it doesn’t exist]

PULL

➔ git pull origin <branch_name> [explicitly fetch the changes from the specified branch name merging into current checked out branch]
➔ git pull [fetch the changes from the projects default branch, merging into current checked out branch]

ADD

➔ git add . [Add all changes/new_files to commit]
➔ git add filename.extension [Add some particular changes to commit]

STATUS

➔ git status [To check changes or updates done in local repo]

STASH

➔ git stash [To stash your local changes]
➔ git stash save “Your stash message” [provide a message to describe the stash for better reference]
➔ git stash list [displays a list of all your stashes with their respective messages & stash ref.]
➔ git stash apply <stash> [By default, it applies the most recent stash if <stash> is not specified]
➔ git stash pop <stash> [To apply & remove a stash from the stack]
➔ git stash drop <stash> [remove a specific stash without applying it]
➔ git stash clear [To remove all stashes from the stash stack]
➔ git stash show <stash> [It shows the files modified in the stash along with the changes made.]

COMMIT

➔ git commit -m “V1 Base commit” [commit changes with commit message]

Design By: Dinesh Agarwal

REVIEW

- ➔ git remote [To know where we are, that's where our push is originating from]
- ➔ git branch [what local branch we are working with, at that time]
- ➔ Git branch -a [list both local & remote branch]
- ➔ git diff origin/main [Tells difference between our local repo & our main repo]

RESTORE

➔ git restore <filename> [command reverts the changes made to the file & restore it to the state it was in the last committed version]

OR

➔ git restore –source=HEAD <file> [command discard any local changes made to the file & restores it to the version present in the last commit]

➔ git restore –source=<commit> <file> [This command sets the file to the state it had in the specific commit]

DELETE a branch

➔ git branch -d branch_name [To delete a branch]

Merge vs Rebase

MERGE (Integrates changes from one branch into another, creating a new merge commit to represent the integration.)

```
**Example 1: Using `git merge`

1. Checkout the target branch (`main`):
  git checkout main

2. Merge the changes from the `feature` branch into `main`:
  git merge feature
```

This creates a merge commit in the `main` branch that incorporates the changes from the `feature` branch.

REBASE (Rewrites the history of a branch by applying the commits of another branch on top of it, resulting in a linear history without merge commits.)

(Rebasing is the process of combining or moving a sequence of commits on top of a new base commit)

```
**Example 2: Using `git rebase`**
1. Checkout the branch you want to rebase (`feature`):
  git checkout feature

2. Rebase the `feature` branch onto the `main` branch:
  git rebase main
```

This takes the changes in the `feature` branch and replays them on top of the `main` branch, creating a linear sequence of commits.

Choice between `git merge` and `git rebase` depends on the project's workflow and the desired history. If we want to maintain the original history of both branches and have clear merge points, use `git merge`. Or
If we want a linear and cleaner history, consider using `git rebase`.

********** **END** **********