EX. No: 2	VERSION CONTROLLING IN SOFTWARE APPLICATION DEVELOPMENT

What is a “version control system”?

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.

Types of Version Control Systems:

- Local Version Control Systems
- Centralized Version Control Systems
- Distributed Version Control Systems

Three important steps of version control:

- **git add** changed files to version control tracking.
- **git commit** the changed files to create a unique snapshot of the local repository.
- **git push** those changed files from the local copy of a repository to the cloud

Check the Status of Changes Using GIT Status

Once you start working, you can use the **git status** command to check what changes are being identified by **git**.

To practice working with this command, use the **terminal** to navigate to your git practice repository:

```
$ cd practice-git-skillz
```

Next, run git status.

```
$ git status

On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Notice that when you run git status it returns: **working tree clean**. This means that there are no changes to any files in your repo - YET.

Next, open and make a small change to the README.md file in a text editor. Then, run the command git status to check that changes have been made to your file(s).

```
git status
On branch main
Your branch is up-to-date with 'origin/main'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
```

```
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

The output from the git status command above indicates that you have modified a file (e.g. README.md) that can be added to version control.

Adding and Committing File Changes to Version Control

To keep track of changes to this file using **git**, you need to:

1. first git add the changes to tracking (or staging area), and then
2. git commit the changes to version control.

These two commands make up the bulk of many workflows that use **git** for version control:

- git add: takes a modified file in your working directory and places the modified version in a staging area for review.

- git commit: takes everything from the staging area and makes a permanent snapshot of the current state of your repository that has a unique identifier.

Add Changed Files Using git add

After making changes, you can add either an individual file or groups of files to version control tracking. To add a single file, run the command:

```
git add file-name.extension
```

For example, to add the README.md file, you would use:

```
git add README.md
```

You can also add all of the files that you have edited at the same time using:

```
git add .
```

Commit Changed Files Using git commit

Once you are ready to make a snapshot of the current state of your repository (i.e. move changes from staging area), you can run git commit. The git commit command requires a commit message that describes the snapshot (i.e. changes) that you made in that commit. A commit message should outline what changed and why. These messages:

1. help collaborators and your future self understand what was changed and why.
2. allow you and your collaborators to find (and undo if necessary) changes that were previously made.

When you are not committing a lot of changes, you can create a short one line commit message using the -m flag as follows:

```
git commit -m "Update title and author name in homework for week 3"
```