**Git and Github**

**Version Control:** Version control refers to a system that records changes to a file or a set of files over time, called "versions." In other words, these versions will help you track changes in your code/project and, if necessary, undo those changes as well. This ability to be able to compare, differentiate, and revert changes between two versions of a particular project becomes extremely helpful when working on larger projects. Larger projects mean more people working on the same code, which increases the chances of a conflict. Using version control, you can easily prevent these conflicts.

**Git:** Git is a version control software that you install on your local system. For an individual working on a project alone, Git proves to be excellent software. When working on a project, being on the same page with your teammates is very crucial to avoid any type of conflict. So, how you will do that. Git takes care of all of that, but the only condition here is that each of your team members must have Git installed on their systems. Git also is known as a distributed version control system, which means using Git, you can push and pull yours as well as others' changes to other people’s machines.

**Github:** GitHub is a web-based Git version control repository hosting service. It provides all of the distributed version control and source code management (SCM) functionalities of Git while topping it with a few of its own features. It is **heaven** for developers — a place where they can store their projects and connect with like-minded people. You can think of it as a “cloud for code."

**Git workflow:** A Git Workflow is a recipe or recommendation for how to use Git to accomplish work in a consistent and productive manner. Git workflows encourage users to leverage Git effectively and consistently. There are several publicized Git workflows that may be a good fit for your team.

1. Centralized workflow
2. Feature branch workflow
3. Gitflow workflow
4. Forking workflow

There are two types of version control: centralized and distributed.

1. **Centralized version control**

With centralized version control systems, you have a single “central” copy of your project on a server and commit your changes to this central copy. You pull the files that you need, but you never have a full copy of your project locally. Some of the most common version control systems are centralized, including Subversion (SVN) and Perforce.

1. **Distributed version control**

With distributed version control systems (DVCS), you don't rely on a central server to store all the versions of a project’s files. Instead, you clone a copy of a repository locally so that you have the full history of the project. Two common distributed version control systems are Git and Mercurial.