Code Standard & Best Practise

An interview was conducted to software developers to understand the means of "workflow" and trying to understand they also operate GitHub from their angles. The interviewers also added to the collected data using their own products to witness themselves how GitHub works. From the tasks completed above, results were obtained, and were set organised as a tool, a tool targeted for any developer wishing to improve. Some of the guides are listed below for better work quality and security:

- **Defining code owners for faster code review.**

"Use Code Owners feature to define which teams and people are automatically selected as reviewers for the repository." It is said, when a developer bumps with a code database, it becomes nearly impossible if match; find a specific code for a certain person. And the reason being is that the codebase will be filled with dozens, hundreds, or more repositories and engineers. Even to its smallest groups, there you will find different code owners.

Also including committing code as an unrecognized author can damage ethical proper workflow. Doing so, open a GitHub account using a wrong or not existing host, e.g., email. The hub will define the account to as "unrecognized author", resulting to a potential client gets difficulties in tracking your work done.

- **Do not leak secrets into source control.**

Things like secret projects, passwords, API keys, you should not ever share them to your GitHub and or code source to any other individual around. Rather it is advertised to use an external code source to store whatever personal project in progress or completed.

- **Archiving dead repositories**

As time moves forward, one gets those projects that are no longer needed, or are not referend to anymore and those projects are known to be "dead repositories''. Dead repositories could be any work you were working on or work you have completed and some of those you do not want any one to view them. Therefor it is recommended to archive them so that now will have access to them but you, especially to work to personal projects you wish to hide. Doing that, anyone who is granted access to your GitHub can view other projects but the archived one's. Wishing to view the project will be declined but forced to read the description labelled about the work.

- **Lock package version**

Every package version done should be locked when stored in your source code for later cause to re-building it again, than there only the owner has the key to unlock the project. Best doing so is to specify the version locked in future recognition, if not than the project is at risk of corruption. Keep the locked package the same version for in case of multiple usage occurs, avoids data loss.

- **Leverage task list**

This feature provides the tool to list out what you must cover in future. It helps you keep track on what you currently are working on and what you have done so far. It is advisable to us it. It also helps a lot when it is a group work as too it reminds what is done so far. This feature also assists in keeping the account active and up to date, so it will not dry out.

(Zilberman, E., *Top GitHub best practices guide for developers*, datree.io, No Publisher, No Location, December 2019, https://www.datree.io/resources/github-best-practices#anchor3)

But not all the time a developer will work alone, sometime forward there are tasks that will require a group and the same group will also require to use the same source code. In that process, there are also ways to improve the groups work quality and ensuring security, and they are listed as follow:

- **Make clean, single purpose commits.**

Here we told programmers can easily lose focus, like trying to fix one problem in hand and as time go's another occurs and are than urged to fix the other one, and so on. To prevent that, a suggestion is made for every project that is currently on table. A small commit should be used to label the project to keep the members on board. With that done, all topics will be relevant to the trend situation.

- **Write meaningful commit messages.**

So, adding from the above stated, not only it will keep the topic on one direction but also it will be understanding other members on what is currently happening. Writing a meaningful commit also helps briefing people on any changes that might have occurred during the period.

- **Removing inactive GitHub members**

In terms of group working, removing inactive members is suggested because the person is no longer hands-on on group works so he/she can be revoked from the group. Doing that is avoiding any possible mean of work exposure, "it's better safe than sorry". (Zilberman, E., *Top GitHub best practices guide for developers*, DATREE.IO, No Publisher, No Location, December 2019, https://www.datree.io/resources/github-best-practices#anchor3)

- **Commit early, commit often.**

Every time you just something, even its portion of the entire deal, it is suggested to commit that work to Git every time. Along the run, that will help you in case if anything were to happen to your device than you know that you committed early. "If you are working on a feature branch that could take some time to finish, it helps you keep your code updated with the latest changes so that you avoid conflicts."

- **Do not alter published history.**

Once project uploaded for public recognition, never attempt to put changes. Doing that may change the main concept of the work to something else, side-tracking it. Doing that, could bring about an influence on other members to do the very same thing as the codes would expose to everyone in view.

(No author, *Best practices for using Git*, DEEPSOURCE, Sanket, No publisher, No location, 8 February 2019, https://deepsource.io/blog/git-best-practices/)

Self-Study Plan:

On this next month starting from today I will be focussing on making my portfolio to a proper manner, inserting all the previous work I have done completed and organising them to a much proper and simple place for a decent viewing to whoever wishes to see. Styling it to be much better presentable, and bit by bit understanding JavaScript and jQuery as I will also be attempting to insert those two languages in portfolio.