Assigment 1

The comparison of the three software engineering tools

The different communication tools on GitHub, such as GitHub Issues, GitHub Discussions, and pull requests, se

[Git and GitHub learning resources - GitHub Docs](https://docs.github.com/en/get-started/start-your-journey/git-and-github-learning-resources)

This the communication or collaboration tools that are used in software engineering when are doing analysis everything have to be documenting and analyse(Perez-Riverol et al., 2016) . These resources are also important for collaborative software projects because they enable the organization and sharing of programming tasks between dif- ferent remote contributors.

GitHUB

GitHub hosts Git repositories and provides developers with tools to ship better code through command line features, issues (threaded discussions), pull requests, code review, or the use of a collection of free and for-purchase apps in the GitHub Marketplace. With collaboration layers like the GitHub flow, a community of 100 million developers, and an ecosystem with hundreds of integrations, GitHub changes the way software is built.

GitHub builds collaboration directly into the development process. Work is organized into repositories where developers can outline requirements or direction and set expectations for team members. Then, using the GitHub flow, developers simply create a branch to work on updates, commit changes to save them, open a pull request to propose and discuss changes, and merge pull requests once everyone is on the same page.

VCSs give each contributor a unified and consistent view of a project, surfacing work that's already in progress. Seeing a transparent history of changes, who made them, and how they contribute to the development of a project helps team members stay aligned while working independently.

GitHub- A version control system, or VCS, tracks the history of changes as people and teams collaborate on projects together. As developers make changes to the project, any earlier version of the project can be recovered at any time.

Developers can review project history to find out:

* Which changes were made?
* Who made the changes?
* When were the changes made?
* Why were changes needed?

 Git is commonly used for both open source and commercial software development, with significant benefits for individuals, teams and businesses.

* Git lets developers see the entire timeline of their changes, decisions, and progression of any project in one place. From the moment they access the history of a project, the developer has all the context they need to understand it and start contributing.
* Developers work in every time zone. With a DVCS like Git, collaboration can happen any time while maintaining source code integrity. Using branches, developers can safely propose changes to production code.
* Businesses using Git can break down communication barriers between teams and keep them focused on doing their best work. Plus, Git makes it possible to align experts across a business to collaborate on major projects.

**Storing data as changes to a base version of each file**

With Git, every time you commit, or save the state of your project, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot. To be efficient, if files have not changed, Git doesn’t store the file again, just a link to the previous identical file it has already stored

Most operations in Git need only local files and resources to operate — generally no information is needed from another computer on your network. If you’re used to a CVCS where most operations have that network latency overhead, this aspect of Git will make you think that the gods of speed have blessed Git with unworldly powers. Because you have the entire history of the project right there on your local disk, most operations seem almost instantaneous.

For example, to browse the history of the project, Git doesn’t need to go out to the server to get the history and display it for you — it simply reads it directly from your local database. This means you see the project history almost instantly. If you want to see the changes introduced between the current version of a file and the file a month ago, Git can look up the file a month ago and do a local difference calculation, instead of having to either ask a remote server to do it or pull an older version of the file from the remote server to do it locally.

This also means that there is very little you can’t do if you’re offline or off VPN. If you get on an airplane or a train and want to do a little work, you can commit happily (to your **local** copy, remember?) until you get to a network connection to upload. If you go home and can’t get your VPN client working properly, you can still work. In many other systems, doing so is either impossible or painful. In Perforce, for example, you can’t do much when you aren’t connected to the server; in Subversion and CVS, you can edit files, but you can’t commit changes to your database (because your database is offline). This may not seem like a huge deal, but you may be surprised what a big difference it can make.

**The Three States**

Pay attention now — here is the main thing to remember about Git if you want the rest of your learning process to go smoothly. Git has three main states that your files can reside in: **modified**, **staged**, and **committed**:

* Modified means that you have changed the file but have not committed it to your database yet.
* Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
* Committed means that the data is safely stored in your local database.

**Installing** [Git - Installing Git (git-scm.com)](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git)

JIRA

CodeChargeStudio

Carryout research on each of the software engineering tools listed below and for each tool:

1. GitHub

2. JIRA

3. CodeCharge Studio

 Briefly describe the key features of the tool that can be used by software developers. Not less than 3 references must be cited in this section. [18 marks]

 Discuss the pros and cons of the tool. [16 marks].

 State the tool’s usage and ratings by users. [6 marks]

Guidance on completing assignment:

- A description must give more detail than a simple definition. Provide references.

- A discussion of the pros and cons must clearly show the pros and cons as separate sections. A table can be used in this section as long as it shows detail and has citations of the references.

- The ratings on tools are a fair indication of how good the tool is. State the usage and ratings.

- Technical correctness (language, referencing in Harvard style).

- Copying and pasting from sources forbidden.