

Git Essay

Whether you are working on a solo project or working with a team, accessing your files and projects can be challenging. If you are doing a job for someone, it is important to have a backup of what you are working on. What if you have multiple machines and need to access the files on all of your devices? Some may use flash drives and save the project to that every time.

With Git, you are able to fix all of these concerns you may have. You are able to smooth out your workflow. Git is a version control system that allows you to track changes to code and different versions. If your code ends up breaking, you may look at a previous version and go back to that. In a team setting this is also very important because a lot of stuff goes wrong in those cases. With that in mind, you can have your own branch to work in separately from the others. In a Git repository, there is a main branch that is the primary branch where everything matters. If you are working in a team, everyone will be working on their own piece of the code and the main is where it all comes together.

There are many different commands with Git. you may be working with GitHub to host your code in the cloud using Git. To start, you have to turn your current project into a Git repository. To do this the git repository has to be initialized using "git init". If there is an existing repository you are trying to use, you have to clone the repository using "git clone". That basically clones the repository to your current machine so you can utilize the awesome features of git.

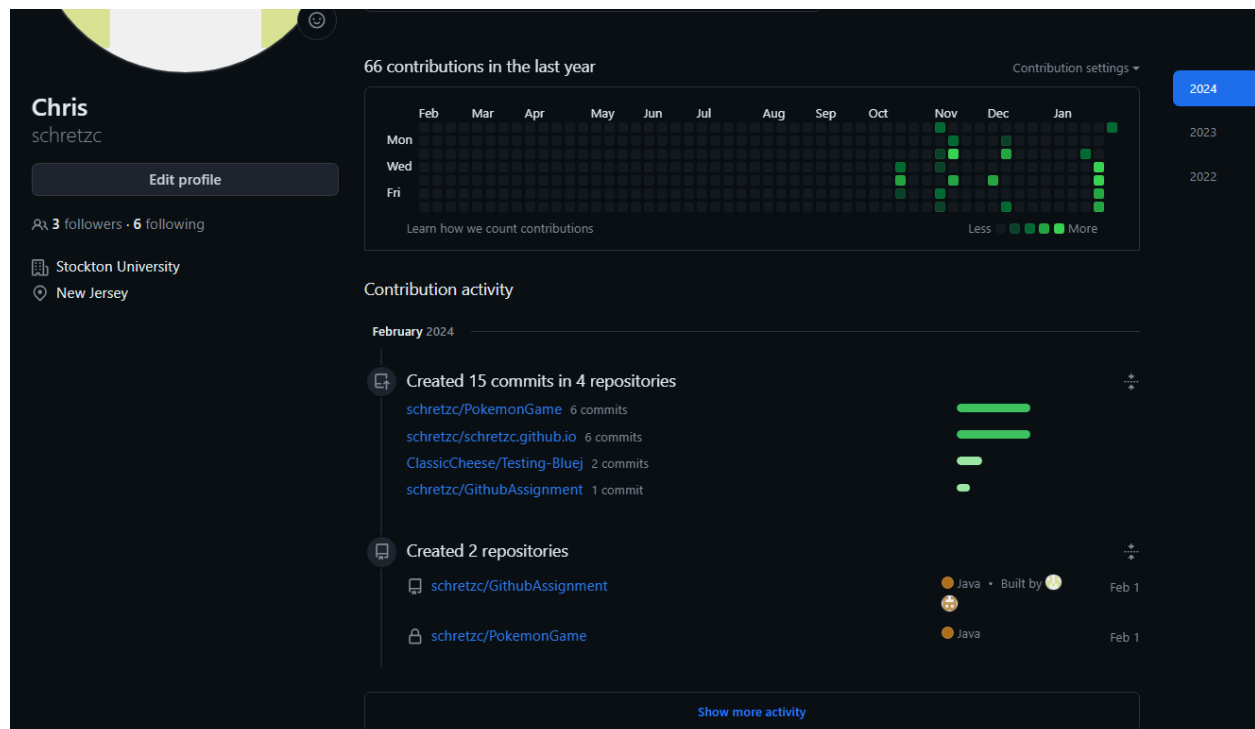
While you are working on your project, whether it is with a team or solo, you have a few simple things to utilize within your workflow. When you want to save what you currently have to the repository, you have to "git commit" your changes. This command saves what you have to the local repository. When you commit, you still have to push what you have to the actual repository. While the concept of git seems very simple, it makes the workflow so much easier, especially with teams. This is used in both small and large settings. The large companies you see every day use this technology as well. When you have branches in a project, each one may have vastly different content. Imagine working on something with other people and everyone is working on different versions of the same project. Sometimes crucial things are not included in someone's current branch. Merge allows you to merge your branch with another branch of your choice whether it is the main branch or someone else's branch. With this in mind, when two branches are out of sync problems could arise. Imagine you are working on a part of the project and in another branch, the thing you are working on is either deleted or

modified, leaving what you are working on to not even exist. This is where a merge conflict happens. The merge conflict does not allow you to merge with that branch and must be resolved for the merge to happen. This feature alone is very useful because the version control software will help prevent really bad compatibility issues from happening or just simply notifying you of such.

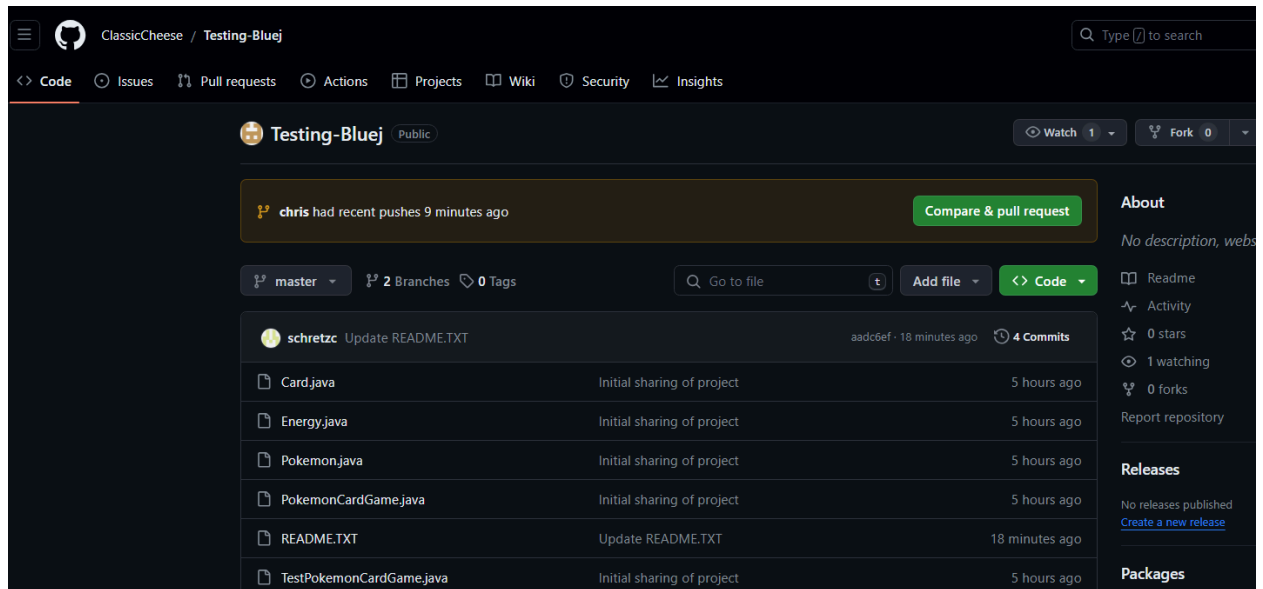
When I was working on a project with my team for the software engineering class, this was the first time I had to work with a team for a larger project. Git allowed us to collaborate on the project ALMOST seamlessly. Some of the people on the team were newer to Git and the GitHub platform and some issues were created. There is nothing wrong with the issues because everyone from my team learned from this regardless if they caused the issue or not. Sometimes they would push bad or the wrong code to the main branch and this caused issues. While we all had our own individual branches, we did not merge that often and basically had different versions of the project in each branch. I think the issue was from being new to the Angular framework for our application. That project was a positive experience of learning version control and git practice but that is only the tip of the iceberg.

There are a lot of useful things to learn with Git and I think it is mandatory if you are going to be looking for a job in this field. There are way more advanced applications of Git that should be learned as well, but the basics SHOULD get you by. Github hosts the repositories in the cloud and the process is usually seamless once learned.

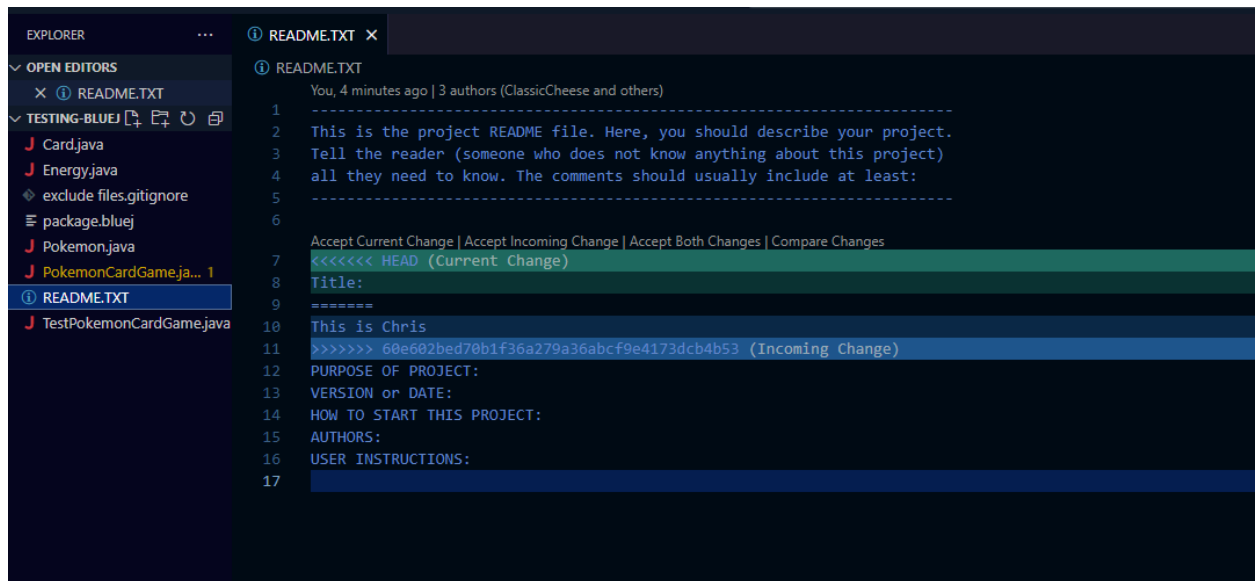
Some code from the class I pushed to GitHub. (was using it prior)



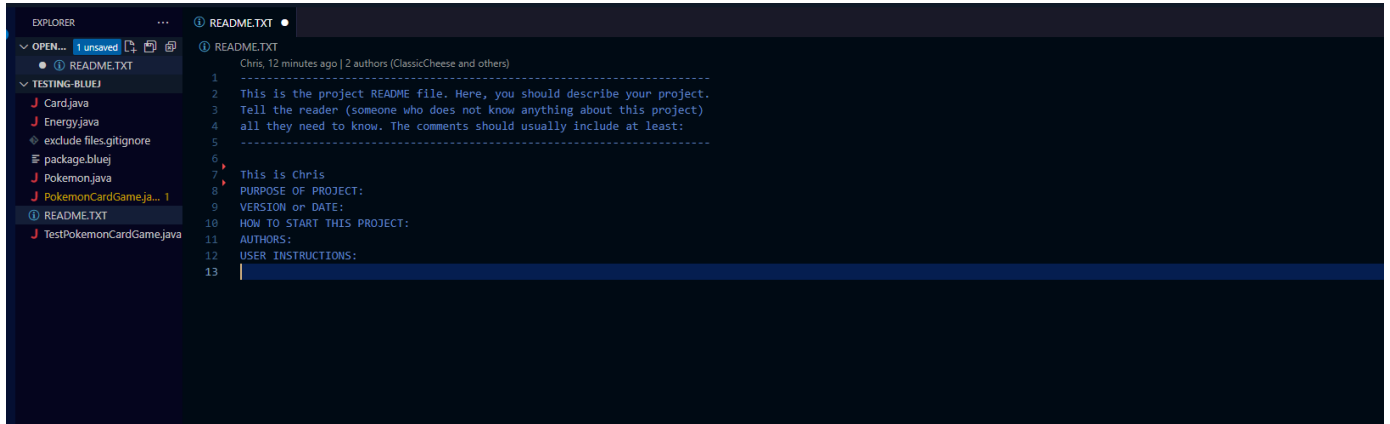
Made change/commit on ClassicCheese's repository



Brownie Points: Forced a merge conflict with the simple text file



Brownie Points: Resolved the merge conflict by replacing the line of code. (I know VSCode makes it way too easy to work with merge conflicts and pretty much everything else once set up properly, but i understand.)



The screenshot shows the VS Code interface with a dark theme. The Explorer sidebar on the left shows a project structure with files like Card.java, Energy.java, and README.TXT. The main editor area displays the content of README.TXT, which contains a merge conflict. The conflict is indicated by a blue line at the end of line 13. The text in the editor is as follows:

```
1 Chris, 12 minutes ago | 2 authors (ClassicCheese and others)
2 -----
3 This is the project README file. Here, you should describe your project.
4 Tell the reader (someone who does not know anything about this project)
5 all they need to know. The comments should usually include at least:
6 -----
7 This is Chris
8 PURPOSE OF PROJECT:
9 VERSION or DATE:
10 HOW TO START THIS PROJECT:
11 AUTHORS:
12 USER INSTRUCTIONS:
13
```