Name: John Burtsche

Term: Fall 2023

Previous Team Projects

I haven't worked on too many projects as part of a programming team before. When I was in Introduction to Computer Science, we had a couple where we would work on a certain program within a group, but most of the time we were just writing pseudocode to base our own projects on and chose the best one out of the group. An example of this was when one of my groups and I created an approach on how to handle a depth first search and breadth first search given certain parameters. Another example is when a group and I determined NP completeness on a particular Travelling salesperson problem. This project was in my Analysis of Algorithm's course, but it was in the same style as Introduction to Computer Science. Besides that, I haven't written any complete programs as part of a group before, so this project provided me with a lot of understanding in how it can work, and honestly how easy it can be. The way that we set up groups was fairly like this class in the sense that it was randomized, but you had the option to select your own group. I don't really know anybody in the online CS program, so I was always randomized.

When it comes to what role I had on these projects…I wouldn't really say that I had a certain role. I feel like most of the time someone in the group typically acts as the leader and I'm okay with doing that if it becomes needed, but will never force that relationship. We are all seen as equals in my eyes and this can create a pretty easy work environment as long as everybody contributes. So far, I haven't had a problem with slackers whatsoever and everything worked out great. The only thing that I will say that I do tend to do, is make sure that everything "looks good" upon completion. With this class project I didn't have to do that, because the flake8 errors were already taken into account so that was nice for my OCDish personality.

When it comes to difficulties that arose, I feel like in my previous groups we never had a problem. Someone's explanation, pseudocode, etc. might not have been up to my standards, but when it came to the grading rubric everything worked out fine so I just typically went with it. The grade reflected that as such too.

The biggest reservation that I have had about group projects is someone doesn't contribute anything. The problems that can arise from this is deciding who has to do the extra work and how to split that up, because you know nobody really wants to do it. I haven't had this problem, so my reservations are getting less and less as I continue through the online CS program.

Working with Continuous Integration

I had a lot of reservations when working with Continuous Integration. This was mainly because I had never worked with it before. When we were setting it up, I was put in charge because I had an account with GitHub premium. To be honest with you, I had no idea what I was doing and ended up having to restart because I thought you had to create the python-app.yml file not build it through the extension in GitHub itself. The videos provided in the module did help a lot and I was able to get a lot out of them to work further beyond the setup as well. My team members were also very understanding and helpful if I

ever had a question or screwed up. Also, every time I was confused on how to do something google had some great videos on GitHub integration (this was the main problem that I had). The biggest problem that I had throughout the whole process was figuring out how to get someone's completed changes to main on my own device (this took me a long time). I eventually figured it out and making pull requests and looking at others became a breeze.

My experience with the mandatory Code Review process went well, especially due to the checks that the python-app.yml made. It got rid of the busy work (mainly flake8 rules and helper functions) so you could just judge someone's code itself in tests.py and task.py. The comments on the code review process were kind of cool too, so people could tell you suggestions and what they liked about it. We had a very active group, so the process was quick and seamless.

I didn't find that we made use of daily commits. We worked well together in our own timelines. If someone for example was enjoying time with their family for Thanksgiving it would take a day or two for the pull request to get merged. I think starting early was the key for us, because it created a stress-free environment for us to work at our own pace, but still get the job done. If someone got something done, there would be someone around ASAP to view the commit, but they could pick and choose when they wanted to work.

As much as I'd like to say that our team followed a Test Driven Development Process… we really didn't. You can say that we performed white box testing, because all of us wanted to test our edge cases and our general code in tests.py, but most of this was done after the code had been written. I would say that the general focus for all of us was to make sure that the code we wrote was correct, and this was the typical style (at least for me) in which I write code then tests afterwards.

I feel like I grew a lot over the course of this project as a developer. I really feel like this is a real-life working scenario where you work as a team with fellow developers. Continuous Integration Workflow was truly an awesome thing to learn because we simultaneously were all able to easily work on a project together. Besides the fact that I learned how to perform this operation, I feel like I learned a lot about being an actual team member. Being there for branch requests, questions and support helped me understand a lot of what it takes for a team to strive. When it comes to growing as a mentor…I feel like we all viewed ourselves as equals so that relationship never existed.

Lessons for the Future

Working with Continuous Integration facilitates better software development, because someone has to check to make sure that what is being added/deleted is correct, while still allowing for team members to contribute when they can. It felt like there was a good amount of safety when it came to breaking the code, but not too much in the sense that it was difficult to get tasks done. I do plan on taking the ideas of Continuous Integration that I learned from this project into consideration towards my future workplace environment.

Mandatory Code review helps individuals become better developers in my opinion because it creates places in which you can comment on the code. If you really liked something you could say it, if you thought something should be changed you can say it to, and your team members can take the criticism into consideration. For example, Lance told me that he suggested adding more tests for function 2 and I

agreed with him and put them in. It really created a no complaints environment towards building the best project possible.

When it comes to a solid test suite when working in a shared repository…it's very important. We took a white box test approach, but once someone completed their code there were tests already available to double check the work in tests.py for the next person. This allowed them to make sure that they didn't mess with someone else's code when they were putting something in. Random backspaces, deletes, etc. are all possible and if you have a test fail when it previously worked you know you messed with another team member's function on accident and can go back a step to find it.