**Name**: Devon Miller

**Term**: Fall 2022

**Previous Team Projects**

I've had several experiences with team projects in my coursework some of which went great some of which caught snags, but I've learned and taken lessons away from all of them. A few team projects I've done have been a large database (databases), a continuous integration workflow (software engineering II), a program to translate dog barks into English (software engineering I) and working as a teaching assistant in Algorithm Analysis. There have been typical challenges of differing schedules, non-responsive teammates and a teammates computer crashed. These initially were roadblocks but the more difficulties I encounter the easier it has become to navigate past them.

My most successful team project was a database written in JavaScript with SQL. This project had weekly status reports and change reports due along with writing the project, we had the option of JavaScript or Python Flask, we chose JavaScript as our prior experience had been primarily in Python and we wanted to learn the most we possibly could. We did have a significant issue when my partner's computer (that he built) crashed part way through, and he wasn't able to write code for a few weeks but was able to write PDF documents. While initially this caused concern our communication was great, and we evenly divided the work with me doing more when he was busy, and he picked up parts of mine the week after. We held weekly meetings where we determined what we needed to do for the week, reviewed what we had so far to make sure it worked correctly, and we had the same vision for the result. While his computer wasn't working well, I wrote the code, and he wrote the PDFs which worked great. Individually I liked this role and was able to do my part with a deep enough understanding to explain the code to him, at the time a new process to me. During weekly meetings I showed him the code I implemented, how it worked, and he showed me what he wrote in our write ups, and we cross checked to make sure we were on the same page. This process was extremely valuable. After his computer was back up and working, he was able to jump in and help fill in boiler plate portions I'd written based on my outline and what I explained during our meetings, then we were able to finish it together and deploy it. The keys to a successful project were communication and understanding what each other had going on outside of school knowing when either of us will need to take on more work and we were both 100% on board in our process and shared vision.

**Working with Continuous Integration**

This course (software engineering II) has been my first experience with continuous integration and my first time making extensive use of GIT in a team setting. This workflow has been extremely useful in ensuring we're all on the same page and if an error has been made it's been caught right away without effecting the version on the main branch. Compared to my previous team experiences where communication was paramount, this process made communication part of the process. Some of the focuses of maintaining code in ways like variable types, layout etc. were handled for us. While extremely useful this cannot replace communication used in previous projects rather it is a failsafe in case anything is missed.

As a team we were asked to each take a function to complete individually and push/pull all functions from the same file as well as having a single test file for all functions. We implemented rules on GitHub to reduce merge conflicts (given the format was meant to create merge conflicts) and took a function as soon as the assignment was made available. In theory having extra time to complete our individual tasks was meant to help a member if they struggle and need help from others. My function took a number of seconds and returned the date as a string determined by the date January 1, 1970, plus the number of seconds taken as a parameter. I broke this function down into two parts, getting the year then getting the month and any days left over were the day in the month. I used test driven development for both of these smaller tasks. When all of my tests for the year returned the correct year and number of days left over with a high degree of certainty, I moved on to getting the month using the same test-driven development process. The rules for leap years made getting the year a tougher task than any other part which is why I started with the year, then I was simply able to merge the parts of my function and wrote new tests to ensure the function worked as a whole unit. I made sure my tests included all edge cases especially on leap years and after leap years.

As a team we employed mandatory code reviews before changes were merged into our main branch which meant code reviews were a vital part of the process. When submitting my code to be reviewed I made sure I was confident my code worked and was free of linting errors, as a reviewer my process was similar. When reviewing code, I used flake8 linter to check for errors then line by line made sure the logic made sense. We used the same test suite for all functions which was useful in ensuring changes to one function did not effect the others and if it did the changes would be caught before branches are merged. When reviewing tests, I took care to ensure edge cases were included along with different input ranges that could cause errors. I made sure to finish my part early so if others needed help, I'd be available for them. Initially this process took getting used to for certain team members and the key was to stay understanding patient and knowing that helping an individual succeed and learn will help everyone in the team. On the less technical side fostering an environment where it's okay to ask questions and the focus is on learning helped us create a successful project.

**Lessons for the Future**

Before this course I viewed GitHub as a place to store my own code, help keep versions organized and safe if anything happens to my computer however this course has changed my view. I will still post all my individual projects to GitHub however when working with teams in the future I'll use this continuous integration process. Hosting on GitHub has helped our members see each other's code without having to wonder how up to date it is and has streamlined our development process. The mandatory code reviews allowed me and other teammates to see how each other wrote code, see ideas to use in our own code and forced us to understand our own ideas well enough to explain them clearly. Looking back this process would've been useful in my database project when my partner's computer crashed. If I knew this process at the time, I would've posted my work after each day or benchmark, and my partner would've been able to monitor it just by logging on to GitHub. Going forward this is how I see myself working on team projects. This new technology has been and will be extremely valuable to my team, however I will remember that communication and collaboration are keys to a successful team. Frequent meetings will still be a part of the process and continuous integration will even help to streamline those meetings by drawing errors to the surface and ensuring we are all able to work to our potential.