Name: Alexander Densmore

Term: Spring 2020

Previous Team Projects

I have worked on team projects in various courses throughout this program. Some projects were focused more on sharing reflections with one another on our learning in the course, such as required group discussions in various classes that allowed us to learn from our classmates' unique experiences and perspectives. Others were focused on software development. I will give an overview of three such projects.

When taking CS 261 (Data Structures), we were required to work in groups to complete each week's required worksheets. While the programming assignments in which we submitted our source code files were individual, these worksheets were preparation for those projects and often involved writing code samples to demonstrate our understanding of various data structures and related algorithms. Since multiple worksheets were assigned per week, my group agreed on a workflow where one person would present the worksheet to the rest of the group and go over their answers and the reasoning for them, and the other group members would then ask questions and provide feedback. Since we knew these worksheets were preparation for the more formal coding assignments where we would be compiling and testing our programs on the school server, we all agreed that, in preparing the worksheets, we would not use a text editor or IDE to write our code snippets and would not test them on the school server. We wanted to be comfortable working with the raw code without syntax highlighting or other convenience features so that we were better prepared for exams in which we would also not have such features available to us. Since our groups were required to keep meeting minutes to be submitted along with our worksheets each week, one member who was not presenting the worksheet would act as the minutes secretary.

My CS 261 worksheet group worked really well together. We were all very engaged in discussion with each of us asking questions or making suggestions about each other's work. We always made sure everyone in the group was comfortable with the answers on a worksheet since each worksheet was just submitted by one person on behalf of the entire group. We got along very well as a group since we all pulled equal weight on the project and all took our work very seriously. When one of us discovered a logic or syntax error in another's work, we approached it as a learning opportunity and gave constructive feedback, being sure to explain the reasoning behind our suggested change. Sometimes we discussed differences in individual preferences to approaching a problem, but such feedback was intended to illustrate for the group another way the same task could be accomplished; it was always up to the worksheet leader whether or not they made stylistic changes based on others' feedback.

In CS 340 (Introduction to Databases), we were required to complete a project in groups of two that spanned the entire term and made up a significant portion of our grade. In this project, we were required to create the frontend and backend for a database service in which users of the website could create, read, update, and delete database entries. My project partner and I completed general planning and documentation together during video calls so that we could ensure we were on the same page with design decisions, and then we each took responsibility for certain pages of the frontend and the related

backend components to drive those frontend pages. We used a shared, private GitHub repository to keep our work in sync. We met once or twice each week to check in, give each other feedback, and discuss our plans for moving forward.

In addition to creating the database service in pairs, each of us was assigned to review other groups' work. We were intentionally assigned to different review groups from our project partner so that each of us could benefit from different students' perspectives on our work. This review process took place once we submitted drafts of portions of the project but before the final versions of those portions were due. We all had to comment on at least two other groups' work, giving constructive feedback. We were then expected to discuss the feedback we received with our project partners and incorporate our peers' feedback accordingly.

Both the database service creation portion and the group review portion of the CS 340 project went well. My project partner and I both had very similar levels of experience and background, so we tended to work at similar speeds and in similar styles. There was one point where my project partner and I were both sometimes running into issues with JavaScript variables declared with "var" and closures. I always used the "var" keyword since we had been advised to use it instead of "let" in CS 290 (Web Development). My partner sometimes used "var" and other times used "let." After doing some research, I realized that "let" variables would work better for our purposes since they have block-level scope rather than function-level cope. I tested whether simply replacing all instances of "var" with "let" would fix the issues we were experiencing, and it did. I then discussed my findings with my partner, and we both agreed to always use the "let" keyword going forward. The group review process could sometimes be slightly more challenging, and this mainly arose from a combination in different levels of experience with web development and differences in the focus of the database services we were creating. However, my review group worked through differences very well. We always aimed to give constructive feedback with reasoning behind any suggested change. We also tried to point out positives even if another group's work was currently not exhibiting the correct functionality. It always helped to put myself in the shoes of others, thinking about the type of feedback that would be helpful for me if I were struggling with the same concept.

A third project in which I have been part of a programming team was the group project in CS 361 (Software Engineering 1). This project involved designing and implementing a software product to meet the requirements of a classmate who was acting as a "customer." The customer did not participate in creating the project but instead provided feedback on whether or not their requirements were being met during the various portions of the assignment. I was both a member of a development team and the customer for another development team. The project explored both the waterfall and agile development methodologies. During the waterfall portion of the assignment, we created requirements documentation, design documentation, and paper prototypes. During the agile portion of the assignment, we wrote customer stories and discussed them with our customer, and we implemented some aspect of the project using pair programming. We did not implement the complete product that we designed since much of the focus of the course was on documentation before beginning implementation, but the two weeks of pair programming at the end allowed us to experience beginning to develop a system based on detailed requirements and design documentation we had created.

Both being part of a development group and a customer for another group went well. In my development group, the main difference of opinion we had to work through at one point was the

handling of the pair programming portion. Due to us being in different time zones and having completely different schedules, one of my teammates did not think the pair programming paradigm of one person coding while the other watches and gives feedback would be feasible. He asserted that we should each simply code individually and then combine all of our work together into one codebase. I sent a follow-up email the day after that meeting explaining why I wanted us all to do pair programming: not only was it a requirement of the assignment, but it would also be a learning opportunity for us. I suggested pairings based on how busy each group member was where each pair consisted of one person with a more rigid schedule and one person with a more flexible schedule who could accommodate the other's rigid schedule. All team members, including the one who originally did not want to do pair programming, agreed to do it in this way, and it worked out well. We each enjoyed and grew from the pair programming experience. My experience as a customer went very smoothly. The group for which I was a customer was very hardworking and sought my feedback at various points throughout the project. I enjoyed having both the perspective of a developer and of a customer during this project.

Working with Continuous Integration

I enjoyed this first experience working with Continuous Integration. I was excited as soon as I was introduced to this concept because I liked the idea of having a linter and testing suite that are run against any newly-committed code. When working on individual projects throughout various courses, I have always tested my work early and often. I like the assurance that my code so far meets requirements before adding additional code so that I do not have the experience of writing all code prior to testing and then having many errors to fix at the end. Since Continuous Integration formalizes this approach, I knew that this approach would be one that I enjoy.

I definitely enjoyed and benefited from Continuous Integration. In addition to the linting and testing of any new commits, the mandatory code review of any new commits was helpful. As a reviewee, I approached pull requests by detailing what I was submitting and what its purpose was. For example, when submitting a refactoring of a function and an additional random test, I explained the purpose of the refactoring and the general functionality of the random test. When one of my functions had a linting error due to being too complex, I left a note with the pull request that I was aware of this linting error and would work on fixing it in the morning but that my teammates were welcome to review what I had in the meantime. When I subsequently updated the pull request by pushing an additional commit where the function was refactored into two functions to avoid making it too complex, I included a comment noting this with the pull request and also notified my teammates via Slack of the update. When I received feedback on pull requests from teammates with suggested changes, I thought about how best to apply those changes. One change that a teammate suggested that at first seemed difficult to accomplish but ended up being a great change was refactoring my random testing so that the helper function which predicted the expected value returned by the function under testing did not replicate too much of the logic from the function under testing but was as independent as possible. I brainstormed various ways and ended up realizing a way this could be accomplished. When pushing the next commit with these changes, I provided comments explaining the reasoning behind the changes.

Having an education background as both a teacher and tutor (I used to teach Latin and I still tutor Essay Writing) as well as a mentoring background as a mentor who provided performance reviews of and suggestions to other tutors, I applied these skills when writing code reviews. I always thought about

how to write comments so that they are constructive and provide reasoning behind the suggestion. For example, in one of my teammates' first pull requests, he did not include any unit tests with his function implementation. I recommended that he add unit tests because it would help verify for all of us that his function behaved as expected and would allow us to take full advantage of the Continuous Integration workflow. He added unit tests as a result.

This Continuous Integration process allowed me to grow in many ways. As a teammate, I practiced resolving differences in opinion professionally, taking the other person's views into account and trying to arrive at the conclusion that would be best for the team. For example, when we had our first meeting to discuss how we would divide up who would be responsible for writing which functions and what our code review procedures would be, one of my teammates stated that he had already drafted two of the three required functions and that we could just use his drafts if we wanted. Since there were three of us in the group, I and the third teammate both expressed how we wanted work to be divided somewhat evenly so that each person could make a significant contribution in terms of code and have multiple pull requests to be reviewed. The teammate who originally wanted to use his drafts for two functions understood our reasoning and agreed, and then we decided which function each of us would work on. Another time when I worked through a few differences of opinion was in response to one of the reviews of my code. The reviewer had left many comments and I could tell that she intended all of them to be helpful. I was happy to change my code for some of them right away, such as refactoring variable names to make their purpose even clearer or refactoring a function to reduce the number of conditionals by just using immediate returns upon certain conditions evaluating as true. There were other suggestions that were much more differences in stylistic preferences. For instance, the group member stated that she felt my code contained too many comments and that she felt code should be written in such a way that it has little to no comments. I responded that I understood where she was coming from and definitely agree that code should be readable with good variable names and function names that are as self-explanatory as possible where they can be understood even without comments, but I explained how I personally find comments very helpful when I come back to work on code and pick up from where I left off. I also find them useful for situations in which the reasoning behind a calculation or conditional is not immediately apparent. Therefore, I indicated that I would be keeping my comments in the code but that I would be certainly keep her advice in mind, and the teammate agreed to disagree on this.

I grew as a developer by learning new skills and strengthening other ones. Although I have used GitHub in the past for collaboration on group projects and for backing up my individual work, I have always used it simply as a place for storing versions of files as I update them. Prior to taking this class, I had never used branches other than the master branch. I enjoyed the experience of creating a new branch copied from master, making changes, and then submitting a pull request with that branch. While our pull requests never conflicted with the master branch, there was one instance where another team member's pull request was merged with master after I had already started making changes in my own branch. Since I knew that this would cause merge conflicts if I submitted a new pull request without first pulling the changes from master, I pulled those changes into my branch. There were merge conflicts, but I worked through each of them, ensuring that I was not changing other teammates' already approved changes but instead was simply getting rid of conflicting information such as two different import statements from the same module or two different function headings. An additional skill that I learned is including meaningful, succinct comments with each commit even if I am not immediately pushing that commit to GitHub. One of my teammates pointed out that the entire commit history on that branch

shows up in a pull request, so meaningful comments on the purpose of a commit as well as deliberate choices for what changes to commit together in one commit help the reviewer. I was sure to be more intentional with my commit comments and what I committed together going forward. I have been able to immediately apply everything new that I have learned about GitHub outside of this class by completing projects in CS 464 (Open Source Software). In that class, our final project includes submitting a pull request to an open source project with suggested changes. I have felt a lot more confident working with GitHub in that class, forking the original repository, making changes, and creating commits than I otherwise might have felt.

Lessons for the Future

This project has allowed me to experience firsthand how useful Continuous Integration is when working on a project. Creating pull requests with new updates to our codebase throughout the duration of the project ensured that everyone's code could be run against the linter and test suite as well as reviewed regularly. Due to including unit tests in our codebase for specific functionality of our functions, we were all able to clearly see at any point what work had already been done and what requirements we could verify had been met. We knew that, if code we committed passed linting checks, we would not need to worry about going back and changing it later to make it comply with flake8 requirements. Furthermore, the test suites against which all changes were tested assured us all that no change we made broke previous functionality, either of our own function or of a teammate's function. Automating linting and testing in a Continuous Integration workflow ensures that all team members' newly-committed code is linted and tested so that no one neglects to lint or test code, and requiring multiple commits throughout the duration of the project assured that no one only had this linting and testing run against their function and tests at the very end.

Individual code reviews were also a great learning experience. I saw how getting timely feedback from others upon making a pull request allowed me to integrate their feedback going forward. For example, one of my teammate's pointed out that, rather than creating a top-level variable to be treated as a constant containing all the hexadecimal digits, I could just use the string.hexdigits constant built into Python's string library. I was previously unaware of this, so I thanked the teammate for this suggestion, refactored previously written code to utilize it, and used it in future code that I wrote as well. It was great that I did not need to wait until the very end of the project's implementation to receive this feedback. I definitely understand how these individual code reviews allow developers to continually grow while working on software.