Team: Discord Dragons
Members: Dana, Kevin, Max, Swamik
Project: 4B.1 - Pair Programming Report
Product Proposal:
Code: https://github.com/Kayala47/SDEV-KEKW/blob/master/code/bot_interface.py
Commit History: https://github.com/Kayala47/SDEV-KEKW/commits/master/code
Primary Author: Dana, Swamik
Slip Days: 0

Pair Programming Report

For our team's pair programming exercise, Dana and Swamik worked on the implementation of the bot interface and its testing suite. This component's primary author was Swamik, so he had the most knowledge about how to implement and test the module. Dana, on the other hand, is the primary author for the initiative tracker module, so she was not as familiar with the syntax used for the interface to interact with Discord messages. As such, the process began with a run-down of Discord syntax, to allow for pair programming to run more smoothly later on.

We thought that we worked well together and divided up the work in a way that the pair programming exercise was most effective. Throughout, we alternated who the driver and navigator was depending on who was more familiar with the section being worked on at the time. In general, the work was divided in a code/review method. The driver would be writing the code while the navigator reviewed each line of code for typos and logic errors. In terms of how the sections of code were divided, Dana typed most tedious segments (not too much thinking involved), while Swamik navigated. On the other hand, Swamik coded more involved segments (more thinking involved), while Dana reviewed the code for typos and potential issues. For more difficult component features, like parsing the strings needed for the roll function, we both spent

some time thinking about and walking through the logic together before starting to code the function.

Throughout the whole process, in general, we felt that both of our times were used effectively. Having another person there to review the code while one person was typing allowed for many typo and small logic errors to be found swiftly. However, for tedious tasks, like writing the test suite and test cases, as there is not much logic involved, one person was just watching the other type. In that case, we felt time was not used effectively. But, the positives of the code/ review process outweighed the negatives of one person being idle for a small period of time.

This process of code development became a lot faster than if Swamik had worked on it alone. The bot interface file, not including the testing suite, includes approximately 400 lines of code, which is a lot to complete within the short time frame if one was working alone. Because we were able to communicate and walk through the logic of hard component features with a partner while programming, we came to solutions much more quickly. In addition, because Dana already knew her way around the initiative tracker module, it was helpful to have her working alongside Swamik when working on that section of the interface module. This gave a better understanding of how to import and call functions from other modules within the same directory. And, when we ran into difficult problems that required research into the Discord API, our research speed, of looking at Stack Overflow, doubled. One person was able to search for guides on correct syntax usage, while the other continued to type up code. This allowed us to find solutions and debug much more quickly.

We both thoroughly enjoyed pair programming together because we got along well and worked together effectively. At the beginning of the process, it was difficult for Dana to get a handle on the syntax used for their bot to interact with the Discord server, however Swamik was

very patient and did a good job at explaining it. This allowed Dana to more effectively navigate later on, which was very helpful and made for an overall more pleasant pair programming experience. On the other hand, Swamik started out not having a fully solid grasp of how the initiative tracker module would work and interact with the bot interface, so it was helpful to have Dana to navigate him through it. Once we reached the code for parsing the functions for the rolling module, however, both parties were confused about how to best tackle it. This caused a lot of set-backs, and deleting and rewriting entire chunks of code. This was the most unpleasant part of this experience, because it took a very long time for either of us to understand how to best go about the problem and ensure the code was clean.

Even though we thought that this process was an overall success, there are certain things that we wish were different. Because this semester was remote, we had to do the pair programming process all online. With the internet being what it is, there was often lag in the voice communication and lag in the computer screen when someone was typing a section of code. As such, the whole process was not as successful and time efficient as it would have been if we were pair programming in person and in front of the same computer screen. As aforementioned, during the times when one person was writing tedious code, the other person was essentially just watching the other person type. Next time, we think that the navigator, during times of tedious code being typed, should also give more thought into what other functions need to be written and what additional information we might need to write those functions. If the navigator can look ahead and do some research, when the driver catches up, writing of the next few functions would have sped up drastically.