Bailee Crandley
Ryan Filgas
Dylan Hanson
Jesse Olsen-Jacobsen
Jeremiah Tilse

Retrospective

Over the course of the term, we had several triumphs and several difficulties as well. Our team works incredibly well and fast under a time crunch, and are able to get a lot of work done in a short amount of time. This is important as most, if not all, of us are taking this class along with a systems course. The flip side of this however is that the blocks of time to do the project increasingly shift towards deadlines. As the term progressed, communication became less frequent as we were all struggling to make it through classes.

Project wise, we were able to pull requirements, design, and test planning off well. Although, our design faced many changes, even up to the programming phase. Perhaps the biggest triumph of this group project was being able to have an open mind to work with others, and a curiosity to learn new things. The three initial phases of our assignment demonstrated these traits, and our group showed incredible persistence as we pulled through the final hours of our project.

Requirements gathering: Members of our team worked collaboratively and openly discussed what the project requirements were. This discourse narrowed down the requirements substantially and spurred questions to ask our professor for further clarification. As a team we each reviewed each other's work for improvements, and the project was better for it. Despite being incredibly thorough, we still had questions about requirements long into the project which was one of the negatives. On the positive side, we were able to look at each other's work in real time on google docs to make comments and recommendations which proved incredibly effective. Appreciating others for their perspective made this a smooth process.

Design: The design process was a long and interesting road which twisted and turned depending on how requirements were interpreted. An initial design was too complicated which led to a redesign a few days before the approaching deadline. This part of the project was larger than that of the requirements phase, and the detailed design took everyone’s help to complete. As usual we were able to pull through. The negative in this case was that more group discourse may have led to a better program design. This can be chalked up in part to busy schedules as the term took a more challenging turn.

Testing: The process of building a test plan was a fast one and contained much of the information from our two previous deliverables. A positive for this assignment was that we had done such a thorough job in earlier work, this became a much easier process to digest. The negative in this case is that it’s hard to build tests amid changing designs and requirements and have them last a useful amount of time. As a result of changing design, some of the testing had to be reworked.

Along with this, an incomplete understanding of unit tests led to an incomplete plan for testing. Most of the unit tests had to be designed on the spot once a more thorough understanding was acquired.

Implementation: Our group worked with git and communicated well to avoid merge conflicts. The design of the project was changed on the fly during this phase in order to reduce work and allowed us to progress faster than we would have normally. In addition we chose a programming language we were all familiar with, but opted to learn parts of modern C++ more thoroughly to expedite the programming process. It was clear while programming that the design had holes in it so to speak and didn't account for persistence of data. In addition, variable names in the program could cause potential confusion as well. We also encountered an issue where different people were using different values to signify success or failure. With multiple people working on different functions and one person working on testing, consistency was mandatory. This is a case where if work is divided such that one person is doing prototypes for another person to program, clear comments in the code are incredibly important. While explaining concepts to each other about the data structure we also found it important to use visual aids when talking about architecture.

Recommendations: Our group could use more collective input on the design phases for the architecture, as more collaboration in this regard would have reduced hours of rework. Starting earlier on the project would have been ideal had schedules allowed in addition to early and frequent communication. In general though, requirements gathering ended up being the crucial piece of the puzzle. Doing this well seems to be something one gets better at over time, and down to the end, the biggest thing that caused rework was misunderstanding of requirements fully early on, but thankfully it was recoverable.

For the next project we would make a few changes. The first of which is to pick a language that has easy-to-use testing frameworks for group projects such as this. Getting a testing framework up and running has been incredibly challenging, but languages like python have unit testing built into the native libraries. Another aspect that we would seek to improve upon is our level of consistency and communication. Throughout the project, we had many bursts of rapid development right before a due date. Although this was effective in terms of getting work done, the quality of the work was lacking, and it added unnecessary stress to our team. Along with this, there were often long dead periods with no communication from certain team members whatsoever. In the future, we would strive for even, consistent development on a project to improve quality and alleviate stress.

Another aspect our group struggled with was the use of modern C++. Using new frameworks, such as vectors and maps, proved difficult to code and compile. We struggled indexing into the map, which sounds rudimentary and simple, but proved to be a laborious process. In addition, a member of our team lost several hours due to those modern constructs being used, as their compiler needed to use the c++ 20 flag instead of the c++ 17 flag. In the future, on a project like this, making use of integers rather than strings as keys would likely ease the use of maps and make compiling easier. Using our current knowledge, rather than attempting too many new things at once, would likely be wise.

Some aspects that we would replicate from this project are collaboration and use of good development practices. Although we lacked communication, our level of collaboration was astronomical. Each team member was open-minded to ideas from other members and never criticized anyone's attempts. Git was instrumental in the coding process, and it's hard to see a way to collaborate on this level without it. After making several mistakes, we have learned an incredible amount about what not to do, and we wouldn't necessarily have gained this knowledge had the project gone flawlessly.