For this project, we were assigned the waterfall development process, which showed us the strengths and weaknesses of developing software using this process. By beginning with documentation, we had a clear goal and understanding of what we needed for our software to perform as expected. This was very helpful because it provided a structured approach that shortened the development time since we did not spend much time developing new ideas or realizing that things we originally wanted to do might not work. However, a downside to this development process was that it was inflexible to change because the requirements, once developed, were set in stone. If, part of the way through the project, we decided we did not need a feature, wanted to add a feature, or change how something worked, we were unable to go back and do so, which would have been extremely frustrating if we had that issue.

Going into the project, our initial expectations for ChatGPT’s performance were low. We had prior experience with it and found it was often prone to failure due to omitting or providing incorrect information. We were pleasantly surprised to find that it generated an SRS document with few errors, which were easily corrected after further prompting. Additionally, we were even more surprised that it gave us compilable code that worked as we expected, with the exception of a few error-handling issues. These issues were resolved through further prompting and our own examination and programming. Ultimately, the outcome of our project was much better than we expected, thanks to our collaboration with each other and ChatGPT.

ChatGPT’s involvement in the project was integral, and we worked alongside it as if it were a third member of our group. It helped us plan out requirements, functions, and classes based on the vague prompt for a study buddy system we were given. It provided us with a working base to use for our project that we could easily understand and modify. It was also beneficial in providing test cases to ensure our code worked as expected without having to manually test every time we made modifications.