The two models used in this assignment were ChatGPT and Gemini. These LLMs were used in three distinct ways over the three sprints: Understanding the process of agile development within the scope of this assignment, templating and refactoring code for specific tasks within each sprint, and developing test cases with respect to the user stories and the code written for each sprint.

Gemini is a key help in understanding how to structure the sprint plans, stand-up logs, and retrospective notes. We did not understand the purpose of stand-up logs and retrospective notes until Gemini provided a clear and detailed explanation of each, with examples to show how we might fill them out. We then applied the templates similar to those that which was generated by Gemini to their respective markdown files. This is how we built the structure to follow during the sprints. However, some of the templates had unneeded fields, such as in stand_up_logs.md, a field for previous stand up's work was provided; however, for the first sprint, there was no previous stand up and adding a field for sprints two and three would yield repetitive statements to the documents.

Templating and refactoring code are done using ChatGPT. We used it to sketch initial module structures (service classes, data classes, CLI command patterns) and then iteratively refine them so each sprint's code matched active user stories in the backlog. Instead of requesting large rewrites, we broke specifics into smaller and focused prompts, which reduced unintended side effects and made code reviews faster. This meant that for the initial generation of code, it suggested consistent naming conventions and separation of concerns. When refactoring, we asked for incremental adjustments rather than broad "optimize everything" passes. This preserved working behavior and any progress regression risk while we continued to inspect each change for hidden assumptions or over-engineered suggestions. This improved speed and modularity while keeping the team's oversight and correctness.

Gemini developed test cases in sprints one, two, and three, respectively, for user stories one, user stories two, and user stories three & four. Gemini's strengths lie in the mass construction of simple requests, such as simple test cases for distinct Python calls to the studybuddy application. However, evaluation of what is generated is a necessity, as some test cases that were generated contained faulty logic, which did not align with how the application was intended to be used. For example, a logical error was found in the test case for user story three "test_search_classmates_in_course" where, after setting up three students' availability, the test logically failed to find a student's availability for the course MATH 2060. This, however, is a simple fix, but it showcases the possibility of hallucinations within LLM-generated code, noting that critical evaluation of such code is a necessity before use.

Overall, the use of AI significantly sped up the process of agile development, leading to an understanding of the agile process without having to focus on the logic of high-efficiency code. It allowed the speed-up of understanding agile development, templating/refactoring the codebase, and developing test cases, which would take many more hours without the use of LLM tools. Such benefits did not come without fault, as with the sped up development process, higher vigilance of code and understanding was necessary to complete the task. The process felt more akin to managing coders within an agile workflow rather than being a coder focused on logical efficiency.