Strengths and Weaknesses of Agile Method

Through our project's agile methodology, we came across various strengths and weaknesses. The adaptability of the process became apparent when our testing revealed problems; we were able to adjust our requirements and backlogs as necessary. This iterative approach also led to quicker product development, and therefore more continuous testing and visual/gradual progress. Furthermore, this early and frequent testing contributed to improved quality, as we were able to identify and fix defects much earlier in the development cycle, resulting in a more refined final product. Finally, the required daily stand-up meetings increased team interaction, which allowed us to discuss any problems we encountered when implementing our features.

While doing these projects, we also encountered some issues with the agile method. Sometimes it feels as if you are implementing features without seeing the whole picture; it seems too short-term focused. Another key point is that sometimes we get keyboard happy and implement too many unnecessary features. Features that were not a requirement, but we ended up wanting to implement because we didn't have a rigid plan. Another issue is limited documentation, since stand-ups are more about communication, it's hard to transcribe everything you state. This wasn't necessary, something that was felt for this scale of project, but with larger projects, constantly having to meet smaller deadlines can become quite difficult or stressful.

Expectation vs Reality

Since I had never used the waterfall or agile processes before, I didn't have a baseline to compare to. However, after this process, we both gained a better understanding of how the software development process works, albeit on a smaller scale. We understood more of the user requirements gathering perspective, as well as making sure not to lose the scope of the project and implement features that aren't necessary. We expected the process to involve more coding; however, we realized that documentation can be more important than the actual code. Our realization is that better documentation can help future software developers make changes to our code, as well as help them understand what our client is expecting from the application.

ChatGPT Analysis

Our use of ChatGPT influenced our requirements gathering as well as our coding process. When gathering requirements, it successfully captured the main points, but we sometimes missed the edge cases. For example, it didn't initially account for the need for "return to dashboard" buttons, which we later had to add to our backlog. For coding aspects, we used ChatGPT to write all of the code due to the short amount of time we had to complete the project. Any issues we encountered in its coding process, usually resolved within the next 1-2 prompts. We realized that sometimes, when you provide large prompts, they tend to miss the details mentioned earlier.