Sprint Review and Retrospective

Angel I. Rivera Perez

CS 250 Software Development Lifecycle

Prof. Jimmy Farley

Date: 2025-12-13

Sprint Review

Working through this project has helped me understand how Agile practices shape the entire development lifecycle and influence the quality of the final product. The SNHU Travel Project gave me the opportunity to step into multiple roles across the Scrum Team, and each role shifted how I viewed the work. As a Product Owner, I learned how difficult it is to express user needs clearly when translating ideas into user stories. As a Developer, I realized how much efficiency depends on having the right level of details before writing code. As a Tester, I saw firsthand how even small gaps in acceptance criteria can lead to confusion about what "done" actually means. Moving between these perspectives made me appreciate how interconnected Agile roles are and how much they rely on consistent communication and shared understanding to succeed.

When reviewing the functionality created during the sprint, I recognized how much clarity influences progress. One example is the work around The Top 5 Destinations slideshow. Running

the JAR file and observing the image transitions, descriptions, and navigation buttons helped me see what a finished increment looks like from a testing standpoint. The slideshow worked as expected but analyzing it through the lens of a tester made me realize how specific the acceptance criteria need to be. Even something as simple as the “Next” and “Previous” buttons need to be tested for boundary behavior, slide order, text visibility, and formatting. The experience showed me how important it is to break features down into testable pieces and how valuable hands-on testing is for uncovering details that may not appear in written documentation. The slide show became a practical example of how an increment demonstrates progress, even if the overall version is still forming.

My work developing user stories taught me how essential it is for the Product Owner to think from the user’s perspective rather than the system’s perspective. Writing high-level stories for different traveler types required me to consider what they truly want, not just what the system could provide. This became even clearer when I produced the detailed acceptance criteria for each story. Breaking them down forced me to examine what needed to happen behind the scenes; such as how recommendations update when a profile changes or how a filter persists across a session. Looking back, the initial versions lacked the dept needed for developers or testers to interpret them consistently. I began to understand why refinement is a continuous activity in Agile rather than a one-time step. You cannot write a prefect user story on the first attempt; instead, stories improve through collaboration and feedback.

Sprint Retrospective

The importance of communication became even more obvious when I reviewed the clarification email written by Brian. His questions highlighted that the user stories I created did not specify enough detail to support accurate test cases. For example, he asked how the top-ten list should be arranged, what content should appear in destination descriptions, and how filtering should behave. These were questions I had not fully considered, even though I wrote the stories myself. This moment reinforced that testers rely heavily on the Product Owner for explicit expectations, and the absence of details affects the entire team. It also demonstrated how collaboration reduces assumptions, because the email revealed that different members of the team may interpret the same story in different ways unless the Product Owner guides the conversation.

Acting as a Developer also gave me insight into how requirements affect implementation decisions. When writing the value statements and acceptance criteria for each story, I had to think not only about what users wanted but also about what needed to be built to satisfy those expectations. The process made me think more critically about the Software Development Lifecycle and how Agile encourages iterative design. Rather than planning everything upfront like in Waterfall, Agile lets you build features in small increments while refining them as you go. This made development feel more manageable and helped me appreciate the value of breaking features into small, testable tasks. It also reminded me that clean, modular code depends heavily on clear requirements; without them, developers risk building something that diverges from user needs.

The tester role was where I experienced the most insight about uncertainty. When creating test cases for each user story, I realized how many decisions rely on the acceptance criteria being specific, measurable, and unambiguous. For example, the story about personalized recommendations required details about how many items must appear, what fields each

recommendation should display, and how updates occur when profile information changes. Without those details, the expected results become guesswork. This made me understand while Agile emphasizes collaboration over contract negotiation; working software and shared understanding are better than holding onto assumptions. Testing showed me that quality is shaped long before the code is written. It begins with how the Product Owner defines value and how the team defines the story together.

Throughout this project, I also engaged with Agile estimation practices, especially Plannihg Poker. Practicing this helped me appreciate the importance of addressing uncertainty early. The technique encourages each team member to assign story points independently, which reduces bias and pushes people to explain their reasoning. Although it can take time, the process forces the team to uncover hidden assumptions and creates a shared understanding of the work. This exercise made me realize how inaccurate estimates can undermine a project's momentum and how helpful Agile estimation techniques are for identifying complexity before development begins. Even in a classroom setting, the value of this practice was clear, since it helped me see how misalignment in understanding can lead to unrealistic schedules or unclear expectations.

Reflecting on the entire sprint, I can see what went well and what I would improve in the future. I felt that my ability to think from the perspective of different Scrum roles improved over time, especially when switching between Product Owner and Tester mindsets. I also learned to appreciate the iterative nature of Agile: nothing needs to be perfect at first, but everything needs to continually evolve. At the same time, I recognized areas where I would do better, such as asking clarifying questions earlier, adding more concrete details to user stories, and considering edge cases during test creation. If I were working on a real Scrum team, I would focus on collaborating

more frequently during backlog refinement and ensuring that developers and testers have everything they need before starting a sprint.

This project ultimately helped me understand how Agile is less about rigid practices and more about communication, adaptation, and shared responsibility. By experiencing each role, I learned how the team depends on shared understanding to deliver value, how communication bridges gaps in requirements, and how iterative work reduces pressure to getting everything right at once. I can now see how Scrum events like sprint reviews and retrospectives create opportunities to pause, evaluate, and improve, which is something I would bring into future projects.