Sprint Review and Retrospective

Kevin Lee Rose

CS-250: SDLC

Southern New Hampshire University

October 2025

During the SNHU Travel project, I took on the mantle of all three core Scrum roles, Product Owner, Developer, and Scrum Master. Each role gave me a distinct perspective on Agile collaboration and reinforced the value of shared ownership. As Product Owner, I focused on translating client goals into concise usable user stories with precise acceptance criteria. One example involved building a slideshow feature from uploaded images. I broke the story down so users could select images, set duration, and preview the final output. That front end clarity helped the team avoid ambiguity and reduced rework during development.

When I stepped into the Developer role, I implemented the slideshow logic in Java with a strong focus on modularity and scalability. I commented the code to support reproducibility and mentoring, every function was built to be teachable and maintainable. I validated functionality through iterative testing, refining logic and confirming expected output. The goal wasn’t just to make it work, it was to make it solid, readable, and ready for future hands-on troubleshooting.

As Scrum Master, I facilitated stand-ups and retrospectives, tracked progress using a simplified Kanban board, and removed blockers, whether that meant clarifying vague requirements or resolving merge conflicts. I kept the team aligned and the sprint moving. Agile’s iterative nature was useful when tackling complex stories. For instance, exporting the slideshow as a JAR file started as a vague requirement. We broke it into smaller tasks, configuring the build path, packaging resources, and testing the executable. Each sprint gave us a checkpoint to review progress, adjust priorities, and incorporate feedback.

Another example was refining the image duration feature. Early testing showed that hardcoded durations didn’t meet user expectations. We pivoted quickly, adding a dropdown menu for per-slide duration selection. Agile’s flexibility let us adapt without losing sight of the sprint goals. The backlog evolved as we learned more, always reflecting the current understanding of the product. Midway through development, the client shifted focus toward health-conscious travel experiences. In a waterfall model, this kind of change could have derailed progress. Agile’s responsiveness turned it into an opportunity. We held a backlog refinement session, re-prioritized tasks, and created new user stories that aligned with the updated theme—such as filtering destinations by wellness criteria or highlighting eco-friendly accommodations. Rather than rewriting core logic, we adapted existing modules to support the new focus. This pivot reinforced how Agile absorbs change without sacrificing quality or momentum.

Communication was a key driver of our team’s velocity, especially when it came to resolving ambiguity before it could slow us down. One example was a targeted email I drafted to clarify acceptance criteria with the Product Owner role. The story in question had vague language around expected slideshow behavior, and rather than guessing, I sent a concise, structured question that asked for confirmation on specific user interactions. That single message prevented scope creep and ensured we built exactly what the client expected. It also set a precedent for proactive communication—short, focused, and designed to keep the team aligned. Moments like that reinforced how intentional messaging can eliminate rework and foster collaboration across roles.

We used a simplified Kanban board to track user stories, tasks, and bugs. Each card had a title, description, status, and assignee. That visual workflow made it easy to spot bottlenecks and balance workload. During Sprint Planning, we estimated effort using story points and committed to realistic deliverables. Daily Stand-ups surfaced blockers early. Sprint Reviews let us demo progress and gather feedback. The Retrospective was especially valuable, we used a “Start, Stop, Continue” format to reflect on our process. For example, we agreed to start using version control branches for each feature, stop merging directly into main, and continue annotating code for clarity. Those insights directly improved our effeciency and code quality in subsequent sprints.

Scrum-Agile proved to be the right fit for this project. Its strengths were clear: flexibility to adapt to changing requirements, transparency through regular check-ins and backlog grooming, incremental delivery of working features, and collaboration fostered by role rotation. We faced challenges, early sprints had a learning curve, and frequent meetings required discipline to stay focused and time-boxed. Without full access to tools like Jira, we relied on simplified templates. But the iterative model aligned well with the project’s exploratory nature and emphasis on working software over exhaustive documentation.

Leading the Sprint Review and Retrospective reinforced the value of Agile principles in real-world development. By rotating through roles, adapting to change, and communicating clearly, I saw firsthand how Scrum fosters resilience, clarity, and continuous improvement. For ChadaTech, adopting Scrum-Agile across teams could unlock similar benefits, especially in dynamic, client-facing projects. The SNHU Travel pilot proves that with the right mindset and tools, Agile isn’t just a methodology—it’s a catalyst for better software and stronger teams.