**Sprint Review and Retrospectiv**e  
  
**Applying Roles**

Over the course of this project, I rotated through multiple roles on a Scrum Agile team: Product Owner, Scrum Master, Developer, and Tester. Each of these roles was vital to the success of the SNHU Travel application. For instance, as the Product Owner, I learned how to create clear and concise user stories that aligned with client needs, such as booking logic or UI usability. When I transitioned into the Developer role, those stories gave me the direction I needed without having to stop and ask what each task meant.

One of the most eye opening roles was that of Scrum Master. Leading daily stand ups, even if simulated, highlighted how a few well structured questions ("What did you complete? What's next? Are there blockers?") could guide progress and uncover hidden obstacles. Even in a classroom setting, the Scrum events felt necessary and effective in creating rhythm and structure.

**Completing User Stories**

The Agile framework, particularly the use of sprints, kept development focused and iterative. Each user story was broken down and prioritized within a sprint. This made the work manageable and progress visible. For example, when I worked on the flight booking interface, having that work scoped into a single sprint allowed me to zero in on front end inputs and their logic rather than get overwhelmed by the full application.

Additionally, receiving and incorporating feedback from earlier roles (like from the Product Owner perspective) improved each iteration. Agile’s "inspect and adapt" philosophy meant no story ever felt final until it worked for the user, which kept quality high.

**Handling Interruptions**

During one part of the project, I had to make mid sprint changes to accommodate a new request. In a traditional Waterfall model, this might have caused chaos, as the plan would’ve been locked in. However, Agile made it more manageable. After a quick mock stand up and backlog refinement, I adjusted the story points, prioritized, and the team shifted accordingly.

This change taught me firsthand how Agile embraces change, not just tolerates it. We didn’t need to rewrite the entire plan. We just adjusted direction. And in software development, that flexibility is crucial.

**Communication**

Throughout the course, clear communication was critical. One of my best moments of cross role collaboration came when I drafted an email to the Product Owner and Tester, outlining what I needed to proceed with implementation. It looked something like this:

To: Product Owner & Tester

Subject: Feature Clarification & Test Case Update Request

Hi Team,

Before moving forward with development, I need confirmation on the expected user flow for the multi booking feature. Specifically:

Are users expected to add all legs at once, or one at a time?

Should error handling occur per leg or only after submission?

Additionally, please confirm if test cases for this feature will be updated in the shared QA doc by tomorrow.

Thanks,

Dominic

Developer

This message was effective because it was clear and action oriented. It helped avoid delays and ensured accountability across roles. This kind of communication is a skill I’ll carry into future projects.

**Organizational Tools**

Agile tools like information radiators and digital boards (like JIRA or Trello) are critical in keeping the team on track. In our course simulations, even simple lists or tracking boards brought visibility to the team’s work. These tools promoted transparency and highlighted when someone was blocked.

Scrum events like the Sprint Planning, Stand ups, and Retrospectives created a structured cadence to reflect and refocus. They were simple, but they worked. Without them, it would’ve been easy to drift off course or repeat mistakes.

**Evaluating Agile Process**

The Scrum Agile method was highly effective for this type of evolving, user focused application. Among the pros, we had:

• Fast feedback loops, which led to quick improvements

• Better team awareness and shared responsibility

• Flexibility when priorities shifted mid sprint

However, there were some cons:

• Time boxed sprints felt rushed occasionally

• Switching roles weekly made it harder to go deep into any one discipline (though this was by design for the course)

Overall, Agile was the right choice for this project. A Waterfall approach would have made iteration harder and risked misalignment with the user’s goals. The Agile method's ability to absorb change and promote collaboration made it an ideal fit for the SNHU Travel app.