Working on the Travel application was a real experience for our team. Switching to a Scrum-Agile approach was definitely a big leap from our usual waterfall method. Honestly, it felt scary at first to rely on sprints, user stories, and so much constant feedback. However, as soon as I got into the rhythm of daily stand-ups and iterative development, it started to feel more natural. But as we got deeper into the process, I could see how these Agile practices were really helping us stay organized and adapt quickly to changes. In this retrospective, I’ll walk through the various roles we assumed, how we handled the development of user stories, the interruptions we faced, and the communication strategies we used to keep everyone aligned. I’ll also share the tools and Agile principles that worked best for us and give an honest assessment of how Scrum-Agile measured up for our project.

**Applying Roles**

One of the most striking parts of using Scrum was how clearly defined the roles were. Even though each of us wore multiple hats at different times, we still tried to embody specific Scrum roles when needed:

* **Product Owner (PO):** In one sprint, I stepped into the Product Owner role. This meant gathering the requirements from the “client”—in this case, SNHU Travel—and turning them into user stories. I found that being the PO forced me to think about the “why” behind each feature request. For instance, when SNHU Travel asked for a flight search filter, I translated their need for a user-friendly search experience into concise acceptance criteria. Knowing exactly what the client wanted helped us focus on delivering real value rather than guesswork.
* **Scrum Master:** Later on, I also acted as the Scrum Master, facilitating stand-up meetings and ensuring we followed our sprint schedule. If someone got blocked—say, a teammate needed design assets or was unsure about an API call—it was my job to jump in and coordinate with the right people to remove these obstacles. During one sprint, a teammate struggled with database integration. By bringing the right developer and the PO together, we sorted out the confusion in a matter of hours instead of letting it turn into a huge roadblock.
* **Development Team:** Our development team members were responsible for writing code, testing features, and bringing each user story to life.Our dev team really came together on this project. Everyone jumped in wherever they were needed whether it was building out the user interface or handling the back-end setup. There was no “that’s not my job” mindset; we just focused on getting things done.These roles might sound simple on paper, but in practice, they created a sense of shared accountability that kept our project on track.

**Completing User Stories**

In past projects using the waterfall model, we’d often gather all requirements upfront and then just build everything in a linear sequence. This time around, the Scrum-Agile approach helped us tackle user stories in shorter sprints. We didn’t have to wait until the end to find out if we’d misinterpreted a requirement or if something no longer aligned with the client’s vision. Instead, we planned each sprint around user stories with well-defined acceptance criteria and prioritized them in our backlog.

For example, one user story asked for “A user should be able to filter travel packages by price range.” We discussed it as a team, estimated its complexity, and agreed on the acceptance criteria: the filter should allow a user to input a minimum and maximum price, and the displayed packages should update immediately based on those constraints. Once we had that acceptance criterion mapped out, the developer got to work on the code, while I, acting as the Product Owner for that sprint, checked in every couple of days to ensure the feature was shaping up correctly. Because we were focused and had a short feedback loop, we wrapped that story up in a single sprint.

**Handling Interruptions**

Of course, no project is ever interruption-free. Midway through one of our sprints, SNHU Travel came back with a major scope change: they wanted an additional feature that allowed users to book rental cars alongside flights and hotels. Under a traditional waterfall plan, this kind of change would have been a nightmare, likely causing a total redesign of our project plan. In our Agile setup, though, we simply updated the product backlog and re-prioritized user stories. We put the car rental feature at the top of the backlog for the next sprint. Instead of derailing our entire schedule, we just adapted. The daily stand-ups were critical here, too, because each day we checked in, we could quickly inform one another about new priorities and address any challenges.

**Communication**

Throughout the project, communication was the secret sauce that kept us aligned. We used daily stand-up meetings to share progress, discuss blockers, and give brief updates. Here’s a quick sample of how a typical daily stand-up message might look in our team chat:

**Scrum Master (Me):**  
“Good morning, everyone! Let’s do a quick stand-up.

* Yesterday, I finalized the backlog updates for the new car rental feature request.
* Today, I’m going to sync up with the dev team to confirm the database schema changes needed for the booking feature.
* I’m blocked waiting on the updated design mockups from our UI specialist. Once I have those, I can finalize the acceptance criteria.”

Sharing these bullet points every day kept the entire team in the loop. Whenever someone mentioned a blocker in the stand-up, we’d address it immediately. This open communication prevented small issues from ballooning into serious problems.

In addition to daily check-ins, we held a mid-sprint review if we felt priorities were shifting. These informal reviews were excellent for re-aligning our approach and making sure everyone was comfortable with the work that still needed to be done.

**Organizational Tools**

A big highlight of the Scrum-Agile approach was the range of organizational tools and events that we used to stay on track. We adopted a digital Scrum board (using Trello or Jira) to visualize our backlog, sprint tasks, and progress. This board allowed us to move tasks from “To Do” to “In Progress,” “Testing,” and finally “Done.” Seeing tasks move across the board was both satisfying and a great motivator.

Each of the core Scrum events helped in different ways:

* **Sprint Planning:** We broke down the highest-priority stories and estimated their complexity.
* **Daily Stand-Ups:** Kept everyone aware of blockers and progress.
* **Sprint Review:** Allowed us to demonstrate our working features and gather feedback from stakeholders.
* **Sprint Retrospective:** Gave us a chance to reflect on what worked well and what we could improve for the next sprint.

All these elements combined to create a dynamic yet controlled environment where we could pivot when needed but still keep our goals in sight.

**Evaluating the Agile Process**

Looking back, the Scrum-Agile approach had clear advantages:

* **Pros:** Rapid feedback, ability to handle changes smoothly, and improved team collaboration. Our short sprints ensured continuous progress and frequent check-ins with the client, so misunderstandings were minimized.
* **Cons:** For newcomers, Agile can be overwhelming. The frequent meetings might feel disruptive at first, and some team members had trouble estimating tasks accurately at the beginning. Also, if stakeholders aren’t available to provide timely feedback, sprints can lag.

Despite the learning curve, I believe the Scrum-Agile methodology was a great match. We were able to incorporate new features midstream and still produce working software at the end of each sprint. Had we been using the waterfall model, major changes like the car rental feature would’ve required a huge shift in scope and possibly led to missed deadlines or rushed deliverables. So yes, the Scrum-Agile approach helped us deliver a more flexible and client-focused solution.