# Applying Roles

Throughout the SNHU Travel project we covered the various Scrum roles to gain a greater and deeper understanding of how cross functional collaboration is the core of the Agile methodology. Acting as product owner showed that sharing an idea and requirements was only the beginning of the role, a true product owner remains involved in the process by receiving user feedback and adjusting the backlog accordingly. Successful product development occurs when owners adapt their products based on what users and the market want, remaining steadfast in initial beliefs and assumptions can limit overall marketability in the end. An example of this was changing the main functionality of the app from overall popular vacations, to vacations of just one type.

The scrum master role focused mainly on keeping the team focused and productive. Planning daily standups, performing backlog grooming, and retrospectives were essential tasks to keep the team moving forward. This role is centered around the ideas of servant leadership and facilitation, anticipating the teams’ needs and potential hindrances. I found that having some technical knowledge was also extremely helpful in this role. Without at least a basic knowledge of development and overall software engineering principles, I would have been rather removed from what my team is actually doing. To me, this lack of technical knowledge would not lend itself to the idea of a self sufficient and cross functional team, and I would be more akin to a middle manager as opposed to the servant leader I should be.

While inhabiting the developer role, our priority switched from overall popular travel to detox and wellness travel specifically. This led me to have to reach out to the product owner for some clarifications, and to communicate with the tester to ensure that time was not wasted and both of us were putting effort towards the same expectations. Ultimately though, this role adapted to changes very easily. A lot of the code from previous sprints was easily reworked to fit the evolving product.

The tester also plays a crucial role in this process and while inhabiting this role we experienced what it was like to communicate with the product owner to get clarification on user stories in order to create relevant and productive test cases. I say that this role is crucial because curating the right test cases ensures that developers are working on the correct things. A developer can waste a lot of time trying to create functionality to complete a test case that was never actually needed, and this shows how important it was for me to properly refine and update test cases. To me, this role sort of serves as a bridge between ideas and communication, and technical execution.

# Completing User Stories

Something necessary for completing user stories was a process of backlog refinement, communication, and collaboration. Refining user stories helped get us from ideas to real world outcomes through the format of “As a user, I want \_\_\_ so that \_\_\_”. As previously mentioned, the tester played a crucial role in this step by communicating with the product owner to remove any vague requirements, then passing that information onto developers via useful test cases to be completed. Agile lends itself to this process by being iterative in nature. This means that baked into this method is that idea that stories will constantly be revisited and updated until completion.

# Handling Interruptions

One rather big disruption (initially), in my eyes, was a mid development shift to a wellness travel application from a popular travel application. Agiles’ small, manageable sprints actually made this disruption fairly easy to deal with and didn’t cause many delays. The culture of communication and collaboration within Agile enabled developers and testers to reach out and seek answers to any questions that they had, allowing us to still complete user stories and test cases during that sprint. This also shows how necessary an involved product owner is, delays would have surely occurred if feedback was never given to the testers and devs.

# Communication

Communication was perhaps the most crucial element of this whole process. Taking something from an idea to an actual product requires communication and collaboration amongst many different roles, whether it be between tester and user, scrum master and product owner, or even developer and product owner. During standups, the format of “What I did, what I’m doing, and what’s blocking me” was so simple but served as a powerful driving force behind our sprints. This method helps keep all team members informed about everything that is going on, ensuring the team stays aligned throughout the sprint. While we ourselves did not use organizational tools made for more clear communication and productivity tracking, we covered examples where other companies did so and we analyzed those benefits.

# Organizational Tools

Effective communication is not solely about the interactions and tones between people. In project management environments, tools like Jira and Azure boards ensure that communication of deadlines and tasks is clear and efficient, and these tools can even be integrated to provide real time updates. They allow teams to visually track sprint velocity, track blockers, and prioritize items all in one central location. These tools can be especially useful while working with multiple teams. During this project we stayed within our own team, but we saw examples of other organizations with massive cross-team dependencies and they found these tools necessary to keep everyone on the same page during development.

# Evaluating Agile Process

I think the Scrum-Agile approach worked perfectly for this SNHU Travel project. These methods promote responsiveness, continuous delivery, and early detection of blockers. The roles within these methods are also very well defined, and each role perfectly fits into the bigger picture.

Compared to Waterfall, I think the Scrum-Agile method was by far better suited for this project. Waterfall is too rigid of a system for a constantly changing environment, and not all of the requirements were even known upfront. The lack of feedback mechanisms in Waterfall would also not lend itself to a project like this.