**Scrum-Agile Team Roles:**

To start this off let us look at how the various roles of the Scrum team contributed to the success of the SNHU Travel project. The roles we had in our team were the Scrum Master (myself), the Product Owner, the Developer, and the Tester. Each of these roles were pivotal to the success of the project. First, the Product Owner was the liaison between the end user and the rest of the team, soliciting feedback from our customers. The Product Owner then took the feedback received and outlined concise and actionable user stories with relevant information and priority. A specific example of this was when the end user feedback came in that they wanted targeted “Hot” deals that were relevant to their travel preferences. The Product owner needed to develop new user stories mid sprint to meet these user requirements and ensure they were effectively prioritized so that the most important and achievable (in the time allowed) stories were the ones that were included in the “final” iteration after the sprint. These stories also had clear definitions of done, which allowed the developer to know exactly what the end goal of each story was and the tester to effectively develop automated testing that would ensure all criteria was met. Without the product owner on the team, we would have likely struggled to have a clear idea of what our customers wanted/needed and would have ended with a product that had less business value than we otherwise did.

Moving on to the Scrum master, the Scrum master was instrumental for the success of the overall product, specifically in meeting end goals, coordinating user story competition and removing barriers to project success. The Scrum master held daily meetings where the team could discuss what had been done the day prior, what was going to be worked on today, and what (if any) problems or roadblocks they were facing. After the meeting, the development team had a clear idea of what was going on with the project and where they were intended to go from that point. The Scrum master worked throughout the day to clear any current or foreseeable roadblocks before they bottlenecked development and was able to facilitate the rest of the development team finding solutions. Additionally, the Scrum master is the one that does the sprint reviews and retrospectives looking for ways to improve the entire agile process within the team, for example, what you are reading now. Finally, the Scrum master’s mentorship helped increase the value of each member involved with the product and in time may improve the team to the point where there is no real need for a Scrum master in the process except for mentorship and fostering of team growth.

The developer and tester roles were the real meat and potatoes of entire Scrum team. I choose to talk about them together as while each had an individual role to play their work overlaps to a great extent and considering them as anything, but a pair would likely convolute their roles. The developer built the user stories into software, making the user story go from an idea to a reality, in this sprint the developer added categories to each trip and developed a quasi-database to manage those trips. The developer turned words on a page into code in our software and functional capabilities that enabled us to meet our goals. The developer also coordinated with the tester in explaining processes and testable criteria to ensure the tester could develop automated testing that would ensure all normal and edge cases are handled appropriately. Through that coordination and the user stories definition of done the tester was able to develop automated regression testing to troubleshoot and remove bugs as the project was being developed. This regression testing ensured that no new bugs were introduced as the project iterated upon itself and helped ensure a timely deployment of (effectively) bug free software iterations. Specifically, the tester worked to build test cases during this sprint for each user story to ensure that when the software was ready it could be fully tested to ensure quality. The tester while coordinating with the developer was able to ensure that each new iteration of the project was only an improvement and did not cause the project to go backwards, without this in process testing model we would have likely had major code refactoring at the end of the project that in effect might have killed our timeline and deadline expectations, ultimately without the tester in the agile process we may have been dead on arrival, not only missing our growth opportunity for SNHU Travel but hurting SNHU Travel’s reputation, possibly irreparably.

**Scrum-agile and the SDLC(Software Development Life Cycle):**

The Scrum-agile approach streamlined our SDLC and ensured that no member of the team was idle or completing the lions share of the work. In a waterfall approach our testing would have not even started until the product was completely built. In the agile approach we were building user stories (based off feedback), developing software, testing said software and overcoming problems in the process all simultaneously. When the user requirements were updated by the product owner and prioritized the developer and tester where immediately able to shift priority and start working on the new stories with effectively no loss in time or effort. All each user story is in a sense is a new iteration of a product so each user story became its own compartmentalized task that the product did not need to function but could be seamlessly integrated into the project without missing dependencies. In this specific sprint the product owner received feedback about the product during the sprint and realized we needed to pivot. The product owner then proceeded to write new user stories and prioritize them, including and considering the dependencies each story would have. From there the development team was able to smoothly transition into completing those user stories without leaving the product in an unfunctional state. The agile approach made it possible to change priorities quickly without putting the project back any significant time because if for example you make ten new user stories for ten new features (assuming no dependencies) it does not matter (in project competition time) which story you work on first, as each feature is an independent object, not an interlaced part of the design that causes each other product to fail without implementation. In this agile process once the minimum viable product was developed, every iteration is an improvement on that product. This gave us a huge advantage in completing the user stories as each user story was just another task rather than an integral part of design which could cause numerous delays if pivoted away from. Another thing that helped us complete user stories was that each time requirements changed, old requirements were no longer necessary so if those pieces of the project were not built then there was no time lost in pivot.

**Changing direction and interruptions:**

As discussed above to some extent when the project changed our agile approach was our saving grace. Specifically, when the project changed from offering deals on niche packages to offering deals on packages that were personalized for the user all the previous work was still useful. All that needed to happen was reprioritization and the rest fell into line. If we were not using the agile user story approach the entire team would have been working on products simultaneously that would have required each other to complete and then be merged and debugged together. With the agile approach each team member was working on one thing that (ideally) only took one shift to complete and could be added to the project after testing. This means that if we scrapped a user story at max only one day was lost in development and building test cases. In the waterfall method we would likely have to refactor entire portions of the code base to meet the new requirements and wait until the product was complete to test if all the requirements were met costing us significant time and resources. In a sense agile development is like object-oriented programming where each object can work by itself in a vacuum and dependencies are built on top of rather than concurrently with each other. The prioritization system of user stories ensures that each iteration of the project will be built upon the last rather than requiring future iterations to complete.

**Effective Communication:**

Below is an excerpt from an email sent during this project from the tester to the product owner:

“Hello Christy,

I would like to seek more information about how I should build the necessary test case for User Story ID 1. More specifically I would like to know the technical details as to what exactly will be implemented and how. For Example:

1 .Will the user be able to decide in their price range selection if the list should ascend or descend when displayed?

2. What will be the exact location of the price setting on the profile page, will it have its own subpage, or will it be a general option available on the profile page?”

As you can see this is a clear and concise line of communication between the tester and the product owner. The tester uses the user stories (built by the product owner) to ask clarifying questions directly to the person that has those answers. By asking these specific questions any obscure detail in the user story can quickly be clarified, allowing for more precise information and continued progress of the project. This is the type of communication that is far more difficult to obtain in hierarchical and undefined owner situations. The tester knows exactly who to contact to receive clarification on their problem, the product owner has actual ownership over the product, so they are in effect the deciding and responsible party. There is no chain of indirect communication where the tester must talk to their immediate supervisor, who must talk to another immediate supervisor, who needs to talk to another team leader, who then needs to go to the developer responsible, who then directs them to the person that directed them to complete this feature. As an agile team any member can reach out directly to the responsible party and receive direct information pertaining to their needs. Additionally, if that process doesn’t work the Scrum master is always there to knock down doors and find the answer for them. Another communication from the developer to the product owner exemplifies how much better communication is in agile.

” Do we want to have to make code changes each time we update a trip, or does it make more sense to allow changes from someone with little to no technical knowledge? If the latter is the case, I think the object oriented/data base approach I have completed makes significantly more sense to implement, and we should likely add a UI component to make user changes easier to accomplish in the future.”

As you can see here, the developer though of something that was not initially considered in the planning process. Rather than attempt to route this communication through multiple people who have no direct say in this change, the developer can suggest the improvement to the product owner and have their recommendation added to the backlog and prioritized effectively. While not every communication will have this level of value to the product, valuable contributions like this could be impossible to implement, let alone get approved in a traditional waterfall approach.

**Tools and principles:**

I personally think the product backlog being openly accessible, the daily scrum meeting and the open lines of communication with responsible parties were extremely valuable and directly contributed to the success of the team. These tools and principles allowed each team member to feel value and allowed them to find ways to maximize their value to the team (another agile principle). When the team members were able to see upcoming requirements in the product backlog and the Scrum Board, they were able to foresee potential problems and work to have them rectified before they bottlenecked the team. Having all the information up front leads to overall transparency of the process while facilitating communication. If team members do not know what other team members are having issues with, or that a problem exists before it comes down the pipeline from leadership there is no way they can help to solve those problems. Specifically, the product backlog led to every team member being able to contribute to the shaping of the product and reduction in unforeseen circumstances. Another thing that directly helped the team were scrum meetings where the team members could discuss anything that might have been a problem for them and solicit solutions without fear of looking like they were incompetent. The Scrum meetings encouraged collaboration which almost always improves the capabilities of a team and fosters mutual trust between team members.

**Assessing Scrum-agile effectiveness:**

I will start off with the negative parts of agile in this project that hurt the team then I will discuss the positive side as there is a bit more to talk about in the latter. First off, the agile approach makes pivoting a little too easy, while this is crucial in some projects, under bad supervision to much pivoting could be a problem in and of itself. I think it need to be clear at the start of a project that pivoting during a sprint should be a last resort. The power of agile allows for pivoting but if for example, the project requirements keep changing because of how easy to pivot it is there is a potential the project itself will never meet any definition of done and will be so bogged down with unnecessary feature and iterations that it will become a grabbled mess that could be borderline unusable. While the cost of an individual pivot may be low in agile, the cost of continuous and repeating pivots still can add up into a very substantial mess of a product.

With that out of the way agile has far more benefits than negatives in most applications. While pivoting is one of its primary weaknesses in the wrong hands it can also be an overwhelming strength in the right hands. Being able to pivot quickly to meet market demands can be the difference between a successful launch and a failure of a project that fails to meet user needs/expectations. Another strength is that team members are valuable and can feel that their contribution is valuable. This feature of agile alone can significantly reduce the cost associated with software development and deployment because it reduces employee turn-over which slows the process down and uses comes with it lost time in the form of spinning a new team member up on the entire project. To follow that, agile encourages transparency, trust, collaboration and ownership of the product, all things that can greatly improve a work atmosphere and improve product development times. Finally, a significant “pro” about agile is that it reduces the need for specialist and uneven workloads that come along with that. In a project you may have need for specialist, but in an agile team those specialists can impart their knowledge to other team members reducing their workload with the introduction of pair programming. An example would be a front-end developer being able to pick up and work a back-end development task off the board, be paired with back-end developer and improve their skill set while still providing value to the team immediately, let alone being able to apply those newly learned skills later down the line when the situation requires it.

Given the above information and how the project was completed I think I can honestly say that the Scrum-agile approach was the best approach for the SNHU Travel project. In a traditional waterfall approach, we may have had to discard weeks’ worth of work when we pivoted to the new “detox and wellness” focus of the project. We also would have needed a larger team of developers to meet the end goals of the project in the time allotted, switching people from developing to testing as the project completed to find, track, and remove bugs at the end. We would have likely needed additional weeks to account for refactoring of the code when we found bugs that were intrenched in our application and likely would have had significant project overruns before the project was completed, missing the launch window that was pivotal for the success of the project. While I still think we can improve upon our agile process, this process was most definitely the best approach and will likely add significant value to our company as we adopt it on a larger scale at ChadaTech.