The different roles in a Scrum team all contribute to the success of a project. The Product Manager was integral in conversing with the customers in order to get a grasp on what the users want in the program, this in turn leads to a good understanding for the developers and allows us to implement the features that are most important to the customers. The testers are integral to ensuring that when we do ship something to the customer, it is functional, and close to bug-free. The scrum master facilitates meaningful discussion and ensures that all members are operating as efficiently as possible.

Using the Scrum-agile approach to the SDLC allowed us to properly prioritize features from the user stories that were most important and implement them first. This allowed us to get out what users care most about and helps to keep the team motivated as they can see their work being useful earlier on in the process. This also has the dual purpose of keeping users happy, since we can ship a product that, while unfinished, is still functional and allows them to at least see that we’re not just wasting time; we are actively working on their project and finishing tasks in an order that we see fit.

Using Scum-agile allows us to refocus the project at any point, as our approach is modular. For example, halfway through the design of the app, we decided that we will be focusing on wellness retreats, as they will be trendy this coming vacation season. For us, this was a simple change, we simply recommended the most popular wellness/spa retreats for customers, while we continued to work on the more advanced features in the background. It was simple for us to shift our focus in order to get a product to the customer quickly, which kept the customer happy, without compromising that original vision. Rather than scrapping all the work we had previously done, we simply shifted the priority to this new task we were given.

Having weekly scrum meetings allows us to constantly be in contact with our team members. This means that any issues we run into can be addressed promptly. It’s always helpful to get a new set of eyes on a problem, as people think in different ways. For example, if I were running into an issue getting a program to compile, I may send an email or message to a developer working on my team, and ask them to look over my code in order to see what errors I may be missing. It may be something as simple as a typo that I’m not noticing, and having a fresh set of eyes on it would help to quickly solve them problem, rather than wasting my time, a coworker may see it in a matter of just moments.

The tool that I found most helpful was the user stories. It was so nice to get a clear concise feature that we wanted to get implemented. I have taken quite a few coding courses here at SNHU, and my biggest issue was always figuring out *what* I was supposed to do. Having user stories made it so that I was never worried about what I needed to do, and instead was able to shift my attention to how to do it. This helped me grow as a programmer, since I wasn’t worried about trying to filter out useless information.

The pros for the Scrum-agile process are as follows:

Works best for more complex projects

Clients are able to get a product earlier, and see it be iterated upon

Dynamic and able to be changed as time goes on

The disadvantages for Scrum-agile are:

It is more complicated to implement than the Waterfall method

There is no designated end

There is less documentation on Agile when compared to Waterfall

I think that the SNHU Travel project was a perfect use-case for Scrum-agile. It was a short-term project that we want to change over time, with many requirements that we don’t need to get out ASAP. Using the Scrum-agile method allowed us to get the customer a product very early on while we worked on new features for the program, and when the customer changed their goals for the program we were able to quickly adapt to their new needs.