SOEN 390 Sprint 4: Retrospective

# Introduction

In Sprint 4, the team focused on implementing the feedback from our TA during the demo of Sprint 3. We mainly put emphasis on the coding aspect of the application; we had to complete or achieve close to completion all of our user stories from the release plan. This indicates that almost every feature from the website has been completed and every feature is intended to be the release version of the website. In addition, a lot more tests in the backend as well as the frontend were completed. Currently, we have 6 people working on the backend and 4 people working on the frontend and everyone tests on their end. However, we have yet to work on the mobile development side of the application.

# What went right

## 1 - Team Management

Our group successfully divided up our deliverables in a just and equitable way. We were able to engage and communicate as needed because of this. Effective team management is necessary for high performance, enhanced output and efficiency, and great flexibility. Improved communication among team members may facilitate future sprint planning and meeting organization.

## 2 - Open Communication

Our team was open to talk the whole sprint, despite our lack of accuracy in forecasting the completion dates of many criteria. Members may use our team messaging channel to send messages for help or clarification, and they could expect a response, helpful or not, to their needs. This is a crucial component of a large group project because it gives students the chance to boost their confidence and sense of self, which are essential for efficiency and performance.

## 3 - Coordination and Organisation

We were able to define the deliverables' deadlines thanks to the Agile technique that we employed. It did not take long for us to realize that setting internal deadlines was essential to finishing our sprint on time. In fact, we noticed that decisions had to be made quickly because certain deliveries depended on others. Furthermore, our team did a fantastic job assigning responsibilities and structuring the information. It appears that a single interface included all of the data needed for us to do our duties. Our productivity increased as we were clear about where to get the necessary paperwork for particular tasks. These aspects couldn't have been better because they closely resembled the planning and execution of real-world efforts.

## 4 - Adaptability and Problem-Solving

The group showed an amazing capacity to quickly come up with answers to unforeseen issues and adjust to new circumstances. Because of its flexibility, the project managed to stay on course even with the changing nature of the development process. The team's resilience and collaborative spirit were demonstrated by their proactive approach to problem-solving, which reduced potential delays and maximized output.

## 5 - Collaboration Between Members

Pair programming turned out to be a very helpful practice. By having two team members collaborate on difficult code, we were able to enhance the software's quality and promote knowledge sharing amongst more and less experienced developers. Fewer defects, a common grasp of the software, and quicker learning for more recent hires were the results of this cooperation. The group's cooperation was excellent. Collaboratively, members from disparate functional domains tackled the intricacies involved in merging the front and back ends. The project's momentum was maintained by this synergy, which not only promoted a strong sense of team togetherness but also made sure that cross-functional difficulties were handled quickly and successfully.

# What went wrong

## 1 - Time Management

Ideally we would have had a meeting and split the work up for the sprint at the very beginning of the sprint, but it took about a week after the sprint start date before we distributed the work. This caused us to have less time to do the actual work. We made a coding deadline for ourselves, however at least half of the implementation which had been intended to be completed was still not done by the deadline we had made for ourselves.

## 2 - Lack of Cohesion

Since we did not design and implement the database tables all together, our team ended up making the tables progressively, and each person created tables that were necessary for their parts. This was alright in the first few sprints, however by sprint 4 we started encountering problems because of this approach. For example, there was a table in the database called Assignments which was made for employee requests, however, this table had foreign keys of type VARCHAR that actually needed to be of type INTEGER, and this caused there to be foreign constraint violations anytime someone tried to insert into the table. Therefore another table had to be made called Request for the employee requests which fixed the foreign key data types. Then it turned out that there was also another table which had been made called submittedRequests which was also intended to be for the employee requests. We had to pick exactly which table we were using but by then implementation had already been done based on the structure of the different tables so some changes had to be made to accommodate which table we decided on and it was a very unideal situation which could have been avoided if we had planned and created the database in the first or second sprint.

## 3 - Code management

Actually with code management we did a lot better than in the previous sprints. Our commit messages and Pull requests messages were standardized well. However we still encountered issues with using feature branches. This is because most of the team was using forks, and making the switch to feature branches would have added extra work and slowed down the team. The time to work on implementation was already so limited so using up more time to switch from forks to branches was not ideal. Also, some members of the team had already started implementing some of their parts by the time we knew we needed to switch to branches, and in addition to that, some of the setup was difficult to redo, and in order to avoid these issues some members of the team stuck with using their fork instead of feature branches.

## 4 - Completeness

Some areas of our website lack a sense of completeness. For example our nav bar is still missing functionality. Not all our pages have frontend designs which complement one another, some pages have a vastly different aesthetic than other pages. Overall the appearance of our frontend could be improved, but is at a minimum sufficient.

# Conclusion

The challenges we are facing will enable us to have more experience and a better idea of how to succeed in future projects. For example, designing and making all the database tables at once in a cohesive manner, prior to any extensive implementation. In the future we would also use feature branches and other code management strategies from the very beginning of a project. We are definitely learning a lot from this experience and growing our skills as a team, as well as individually.