Challenges and Lessons Learned

Team CtrlAltDefeat - Brittany Nall & Melissa Smith

CS414 – Fall 2018

# Challenges

1. Collaborating and Communication with Distance.

When collaborating with a team in person, it is easier for details not to get missed and keep everyone on the same page when it comes to what is being done and what needs to be done. When communication is restricted to text messages, emails, and commits to a central repository, it is difficult to keep the team in sync. Face-to-face communication often allows for clearer communication and better collaboration.

1. Time Management.

We found that due to our mutually hectic schedules, it was difficult to manage time to collaborate over Video conference and reply to text messages and emails in a timely manner. It is hard to stay on the same page when the communication turnaround time takes a long time or the other team mate never responds.

1. Clear Roles.

A few times, both team members worked on the same features and wasted time putting effort towards the same functionality.

1. New Tools.

Learning new tools was also a challenge since we had to learn how to effectively integrate Xstream and WindowsBuilder into our project so we had the ability to develop a UI and persist data. Learning these tools increased the development time since neither of us had experience with these tools and we had to learn as we implemented them into the project.

# Lessons Learned

1. Establish clear roles on the team and clear expectations from the beginning. This will help with better management of the team resources and prevent wasted effort by creating duplicate functionality.
2. Keep clear lines of communication between the team members.

This will again help with managing resources and ensuring less time wasted. By addressing these issues, it will be easier to clear up any misunderstandings in discussions of the project.

1. Ensure ideas and planning are clearly discussed as a team.

This will make sure that both of us have enough information to effectively complete your task by engaging in more brainstorming and generating ideas together on how the system should be designed.

1. Extensive testing should be done.

It is important to have someone besides the original developer look through the code and test the system as a whole. If the program runs smoothly on the original developer’s system, it is not guaranteed that it will perform as expected on a different system. It is also important to test part of the system before testing the entire system.

1. Plan for things to not go according to plan.

Even simple changes can cause a ripple in the system that can cause unexpected behavior. Even for a programmer who has been working on the system from the beginning, it can’t always be known how a change will affect the entire system end-to-end. There are just some things that don’t always go according to plan or weren’t clearly defined or thought of during the design phase.

1. Not being afraid to implement new software.

When deciding on what tool to use to implement our UI and store data, several options were looked at. It was difficult to determine which would work best for our project’s needs. XStream met that very well allowing us to store data in serializable xml rather than using our second option MongoDB. Using a database was a little bit more complex than we needed and would require more steps to implementation that we didn’t fully understand. However, while the WindowsBuilder tools provided us with a good way to implement our GUI, it might have been a better option to explore using Javafx. Though our choice suited our needs and provided us with an easy UI, it might have been more worthwhile to implement Javafx as WindowsBuilder is a bit outdated. Javafx would have allowed us to create a more streamlined and sleek UI with more customizability. Going forward, it could have been beneficial to spend more time exploring the different options available in order to create a better finished project.

1. Refactor early.

It’s easy for code to get disorganized and all over the place during the initial implementation. Therefore, it is a good practice to refactor before adding new features.