Challenges and Lessons Learned

Team CtrlAltDefeat - Brittany Nall & Melissa Smith

CS414 – Fall 2018

# Challenges

1. The challenge that arises on every project is collaboration and communication. Even when collaborating with a team in person, it is very easy for details to get missed and not every one is kept on the same page on what is being done or needs to be done. That difficulty is only increased over distance when communication is restricted to phones, emails, Skype calls, and commits to a central repository. In person bandwidth often allows for clearer communication and netter collaboration.
2. Time Management is another common challenge with projects and was true for this as well. Working on the project we found due to our mutually hectic schedules was difficult to manage times to collaborate over Skype.
3. Being clear on what each person has worked on and changed, a few times both of us worked on the same things and wasted time putting effort towards the same functionality.
4. Learning new tools was also a challenge as we had to learn how to effectively integrate stream and WindowsBuilder into our project in order to facilitate our project. Learning these tools had a learning curve that took up development time as neither of us had had experience with these tools before implementing them into the project.

# Lessons Learned

1. Establishing clear roles on the team, and clear expectations from the beginning. This will help prevent with better management of the teams resources and prevent wasted effort by creating duplicate functionality.
2. Keeping clear lines of communication between the team members. This will again help with managing resources and ensuring less time wasted. And determining as a team specific issue which can affect communication in our case distance and the inability to meet face to face. By addressing these issues it will be easier to clear up any misunderstandings in discussions of the project.
3. Ensuring ideas and planning are clearly discussed as a team to ensure 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 as a whole before beginning the true planning & requirements process.
4. Regardless how through a team is on the design and requirements process it is very difficult to plan out every detail of a software project which is why it is often better to start smaller version of the end software and slowly extend it, changing one thing at a time. This helps prevent problems further down the road and allows for smaller iteration and more confident changes.
5. Test, as even with extensive testing done by the original developer it is always better to have someone else look through the code and test the system as a whole. Because even if the program runs smoothly on the original developers system that is no guarantee that it wont perform as expected for someone else on a different system. This means trying to think of every eventuality that the program might encounter and once the code is in place trying to find ways to break it. This also means testing the parts before testing the whole saving time later.
6. Everything take longer than expected and to plan for that. Even simple changes can cause a new unexpected behavior requiring time to determine why the program is behaving that way and what the small change affected. Even for a programmer who has been working on the system from the beginning cant always predict how a change will affect the program as a whole.
7. There will always be unexpected bugs, every project I’ve worked on has reinforced this and the gym management system was no different. Bugs show up everywhere. There can be several problems masked by one bug or multiple bugs that stem from one problem.
8. Not being afraid to implement new software. When deciding on what tool to use to implement our UI and store data I looked at several options. It was difficult to determine which had the best cost to benefits ratio and would work best for our projects needs. Understanding that with anything we decided to implement there would be a learning curve that would cost us time we needed something that was proven, had been used before and could implement the functionality we needed. xStream met that very well allowing us to store data in serializable xml rather than using our second option MongoDB that was a little bit more complex than we needed and would require more steps to implement. 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 a easy UI it might have been more worthwhile to implement Javafx as WindowsBuilder is a bit outdated and Javafx would have allowed us to create a more streamlined and sleek UI with more customizability. Going forward I believe it might be more beneficial to the overall project to spend more time exploring the different options available in order to create a better finished project.
9. Refactor early. Its very easy for code to get haphazard and all over the place. So it is a good practice to make a change, refactor as needed then add a new feature.