SQ-I Semester Debrief

# Hi everyone!

To start with, I want to say that there were some great demos this week. Even for groups that hit some challenges, there were some nice things delivered despite problems. In particular, I’d like to single out 2 teams for really exceptional reasons.

**The Wookies** – Really decent work on the project, good deliverables AND a full user’s manual printed and delivered to both Sean and I. Nice touch!

**Team Green Apple** – Simply the best, console based UI either Sean or I have ever seen in SQ-I. You’d almost swear you were looking at a GUI ☺

There were a lot of other nice touches groups provided in their demos, but I wanted to highlight these two teams for earning the highest levels of ‘I’d almost but this!’ out of Sean on Tuesday.

While I work to finalize marks for this semester, I wanted to share some of the ideas and feedback shared with me from your individual reports, and from the demos Sean and I saw on Tuesday.

Before I begin, if any comment or thought I share below seems to be too much like a comment you made to me in the individual report, please do not worry. The ideas I am sharing below were reported by more than one team or person, and actually fit perfectly with my observations of how SET students deal with this course over the past three years.

# The Technical Stuff

1. **We missed a requirement!?!?!**

Yep! Attention to detail is crucial, and it is hard when you are faced with the kind of requirements documentation we provided you for this assignment. Ask questions. Challenge assumptions. Try to really visualize this impact of these requirements as the end-user of your product would. If your team missed a couple of key requirements (e.g. SIN check digit/validation) you’ll find this wasn’t fatal to your mark, but do commit to a deeper review of the requirements and considering asking more questions of the industry rep in via email or through any Q&A session Sean might offer.

1. **Integration Testing**

I wanted to make a joke here about how in Russia, integration tests you. It really isn’t a joke, because it was soooo true this year. Here’s what I mean – If you thought integration testing only applied to how your software modules behaved when calling each other, you were missing something. Integration is something your group, as a team of software developers must achieve as a set of shared work ethics and communication skills between each other. This means learning to work well, even while being critical of each other. This means using agreed upon tools and methodologies. This means collaborating on good planning and design decisions, like hammering out class responsibilities up front. This was one of the most common things mentioned this year and last… ‘I hope our team can collaborate better next semester’.

One thing that might help was implemented by 2 of our groups this semester – Craft for your team something like an API reference guide at the beginning of the project. Confused as to how our modules should be talking to each other? Unclear as to which methods and parameters are not private? Check the API reference the team drafted! This is one good example of investing wisely in planning.

1. **Our UI was referred to as ‘brutal’ and/or ‘punishing’ to the end user.**

This semester, having a brutal UI was not as impactful to your grade as it might be next semester. It doesn’t take a full degree in User Experience (UX) design to think like the end user. Consider, if it was your job to use this software, would you be upset if one little typo forced you to have to re-enter all the rest of the employee’s record? What about some conveniences for those who have to search the data file to modify one user record?

1. **If only we had done more testing!**

I hear you! Been there… done that ☺ Now that you’ve seen the impact of testing, be prepared to expand your testing skills and coverage in SQ-II. A couple of teams this semester mentioned to me that they want to follow a TDD approach next semester … I think that is a very encouraging idea, and I totally support it!

# The Team Stuff

1. **Your code is out of there!**

Several groups shared that at some point they had to pull one member’s code out of the project, severely change it, or just not implement it. Ideally, we don’t want to be in this situation, but it does happen in the field, so what to do? Firstly, I want to recommend that people remember this was a team project. To an extent, egos need to be put aside in the name of the shared goal of everyone on the team to succeed. I am not asking anyone to cast aside pride in their own work, or to walk away from their professionalism… this is about supporting the team, and realistically facing the prospect of the project being better for not having your code contribution in it.

This is a wicked tough situation, I will grant you. You may feel picked on, you may feel like a jerk for saying ‘Hey Russ… your code base here isn’t working and we don’t have time to work your code into shape.’ Would you tell a customer that you failed to deliver the contract because you felt Russ’ feelings would get hurt? I hear some people saying ‘This is school though. Don’t I have to be marked on code or some kind of contribution to the project’? I would agree, and let me offer this idea… you have to pull Russ’ code, but keep it around. Put it in repo for R&D or test bench code that could be useful to look at or rebuilt at a later time (or to provide to an instructor for ‘proof-of-work’ if needed).

The best defense here is to invest more in planning and pre-emptive measures to avoid having to pull someone’s code at all. Remember how we discussed the costs of software quality and how you can either invest in better controls at design time, or you \*will\* have to invest in repairs, bug fixes and endless cranky customer support calls once your buggy software hits the market? The concept is similar here – Invest up front into establishing clear criteria for success that everyone on the team agrees to meet, along with actions that will be taken if the code doesn’t meet that criteria. ‘Russ, we have to pull your code now, and I hope you will agree based on our team goals we all agreed to at the beginning of the project. We all agreed that our code would be completed on time, be properly encapsulated and have suitable unit tests at this milestone.’ Russ might not like to hear that, but with luck he’ll think about the team’s success and say something like ‘Apologies for that… I thought I was on the right track, but I wasn’t. What can I focus on in the time that is left to help contribute to the team?’

1. **A team member is out of his/her depth in this project.**

You will work with people of all kinds of different skill levels in the industry, and you don’t always get to choose who you work with. One thing I want commend many groups for this semester is the mentoring that occurred in many cases. That’s the best thing you can do for someone who is genuinely trying to contribute but might lack the experience of more seasoned group members. There are some names I’ll always remember…. People who took time to mentor me, Doug, Bill, Gwen, Cindy, Shawn… some guy named Norbert, and so on. Their help to me, is part of the reason I teach and pay it forward.

1. **We had a team member go absent on us.**

This can happen too. Your team should have targets for things like 2 missed communications or check-ins. It is vitally important that you speak to your instructor very early on when agreed-to communications are being missed. We don’t want to play the role of ‘Human Resources’, but we will coach and support your team to try to get things right, but if people are genuinely not participating, we should try to limit their impact to the project as early as possible. For example, I think a few groups in the past few years, including this semester, where this has happened would have preferred dealing with the extra workload earlier than in the last 2 weeks of the course. Communicate early and often and you’ll be more successful.

1. **Time Management**

I know, right? For many different reasons this is one of the hardest things to master throughout one’s life. We always face competing priorities (and things we think are priorities… Fallout 4, why do you haunt my dreams!) but it is generally true that we can always benefit from more work on our time management skills. Although we do tend to get one or two groups in SQ-I that comment about how good it felt to be prepared 1-2 days in advance of the demo, this is rare. It is far more common that we have teams saying things like ‘I wish we had managed our time better!’.

You will get a chance to work on a project with similar scope and time requirements in SQ-II next semester… how will you adapt and develop as a professional software developer to meet the challenge now that you likely know what kind of a time commitment it takes?

Well, that’s all the old-wise-guy comments I can offer you for now. It was great working with you all this semester, and I’m looking forward to having you for Graphics, Animation and Sound next semester! (Yep, that’s a change I just learned about!)

Happy holidays… have a great, well-earned rest.

Cheers!

Russ