Sprint 5 Team Report

Team: 208

Repo URL: https://github.ccs.neu.edu/cs5500/team-208-SP19
Reviewer: Michael Weintraub

Score: 95

Comments

Overall, good job on presentation. Good slides. Could be improved if made a better connection to the course content.

As a pointer, you don't have to provide a detailed outline necessarily. You can be just as successful with a higher-level description of the outline, even if the outline on the slides is detailed.

You shouldn't embellish on the claims. For example, how can the team claim 99+% uptime? No data were given in support of this claim and the method of measurement was never described. You need to be careful with such claims. If it's an optimistic interpretation, you run the risk of losing the trust of your audience. Is this claimed because your deployment didn't crash in the ~4-6 weeks you had it available on AWS? Is this a continuous measure? Did this number include maintenance intervals? How many users were on the system during that period? In other words, did it not crash because it's good; there weren't many transactions; the transaction set exercised a limited functionality set; some of the above; all of the above? In this case, uptime claims usually involve MONTHS of data.

On the feature list, the presentation simply reads the list. In situations like these, think about value add. The audience can read the list, faster than you can talk through it. It's more engaging and helpful if the voice track brings something to the table other than reading the list.

On security, how is TCP a security thing? The payloads are encrypted? Did you mean TLS?

Very good code standards discussion. Everything was justified.

There was good discussion of communication and a nice overview of the sprints. The material would be better buttressed if Jira charts were used. Liked the summary of each sprint. Could have explained more about the lifecycle.

It was a good discussion of the design/architecture. Architecture seems limited by how the proxy is defined, but like that it's an adapter/B2BUA. It's unclear how state is maintained. I suspect each call is blocking, which is fine to start, but not scalable in the end.

Why is dependency injection mentioned as a design pattern?

In the project, the main difficulty was with the REACT routes, but these difficulties were glossed over.

The team produced the Best Demo Video ever! Nice job.

System set-up wasn't really a how-to on setting up the system.