**Presentation (my Notes)**

Question to group: Who here loves to write and maintain tests?

Response: Ok, I think most of you are lieing. I do not believe most people like to write tests. I don’t like writing tests. More specifically, I do not like writing tests are brittle and hard to maintain. But if I have to write tests, I want to write the least number of tests I possibly can that give me the best bang for my buck. I would want these tests to be:

-Easy to maintain (don’t break every time I look at them)

-Fast (so I can run them all in the time it takes me to eat a cookie)

-Great documentation of the systems intended business behavior, not necessarily some obscure implementation of some weird pattern that someone who no longer works here implemented.

-Refactor friendly, which is to say the tests do not concern themselves much with how I internally design my system. They only care about how I satisfy domain use cases.

So how do we get these types of tests?

* We treat testing as a FIRST CLASS CITIZEN. This means we think about testing at the very start of a project, even in the \*gasp\* design phase. Not as something we tack on at the end with little thought or in a reflexive mechanical way.
* We think about our system from a domain perspective. We then also proceed to align our tests to natural domain ‘seams’. Based on how large our system is, this may be one or many.
* We start by making sure we can easily mock out or stub all external process dependencies (databases, apis etc). If these are not easily mockable then that in of itself is a problem that should be addressed architecturally.
* We push all external dependencies to the edges of our system as much as possible. We then create extremely thin interfaces to wrap said dependencies so we can test as much of our code as possible with the least amount of mocking/stubbing possible.
* We supplement our main test battery with low level unit tests *where it makes sense* which should be fairly rare (use game copy example from tictactoe tests). We do the same thing with integration tests (say some very lean tests that make sure that our SQL actually works against a database).