## House Keeping

we will test all non-trivial functions

- triage if needed

- erring on the side of redundancy

how our git will be laid out?

>Master

>Dev

>Testing

Risks

* TA suggested more testers
  + We will consider that later as needed
* Server Testing will have a learning curve
  + But we are saving this as last in priority queue

Smoke test

* Does the Current Build work?
* Should be automated
* Will outline this in the testing plan once we have a design

integration tests

* Each object will have an associated testing class that tests all non-trivial functions

regression tests

* Done by the developers through use of asserts and exception handling, so we shouldn’t be touching this, but we should keep an eye on them and make sure that we don’t have anything to add

User testing

* Can we have users that are not in team 3 test this? We have signed no NDA.
* I don’t think that this will be something that we need other people for, basically we will know if it works or does not work
  + This can be decided later on
* Who will be doing the user testing? Us? Dev? Everyone?

pair programming

* We can do it, but pair testing with the oculus might be more proficient

logging framework

* UberLogger – best documented
* The is also unity default

Logging

* Will be included in testing doc as to what we need to log

style checker

* Whatever dev team decides upon. As long as tabs are 4 spaces.

TDD of Integration tests

* We can be writing tests without having the code in place
* We don’t need to have tests in place before Dev codes

Asserts

* Use em!

Exceptions

* Monitor Devs code if there are any cases that you believe need an exception

\*DO NOT TOUCH DEV CODE

If above situation arises pick a dev and do paired programming with them.

* Hopefully Devs will do the same

Mocks

* Get these all in place real early – they should be the first thing we do before we start writing any integration tests.
* These will be reliant on the Devs Mocks, so they must Mock before we do

Comments

* Once a coding style is picked I will consider a header commenting style that we want to use if we do use one

Testing document

* crc cards
* Descriptions, Significance, Testing Plan
* testing matrix
  + I will maintain this with updates from the testing team
* "diagrams showing path coverage - whether at the level of testing higher-level features or functional blocks in the entire system or at the code level"

peer reviews for particularly tricky bits

* Look for at least 1 peer review per deliverable

testing framework

* unity has one built in that looks excellent

continuous integration

* work with Ix

activity logs

* do them
* I will consider if the testing logs need to be different from the dev logs

how to deal with writing tests for code that’s already written, but not passing the test?

* COMMENT OUT!
* do not commit broken code, but we still want your test in there. Then do paired programming with a dev to uncomment the test, resolve the issue, and then commit

how work will be distributed

* I will probably make some kind of a list after design that you can basically pick an object and start coding, this will likely just be the testing document

write tests for all bug reports

* probably review bug reports because they got through our tests/ we can learn from them

## What’s next?

Mini Milestones:

01-22-17 Requirements Doc First Draft

01-24-17 Architecture and Design Specs First Draft

01-28-17 Testing Plan First Draft

Task List:

01-20-17 Testing and logging framework decisions

* Detail which defect database, testing and logging frameworks (and code coverage/mocking/style checking if using) will be used for the project and why

01-23-17 Implement frameworks

* Set up and configure infrastructure used for testing etc.

01-28-17 Testing plan

* Have the testing plan drafted and ready for inspection