## Unit Test Architecture and Strategy/Framework

Our application service layer is implemented as a Spring Boot application and makes use Spring IOC and its convenient dependency injection in conjunction with JUnit to implement Unit Tests.

Spring allows us to inject references to \*implementations\* of our service and data layer objects when writing unit tests. In practice this gives us a flexible level of indirection, such that we can selectively choose either to inject instances of our production code, or mocked implementations when it desirable to have predictable state / behavior when writing a test.

For our frontend of our application, our main form of testing was User testing. We had our team members go through features implemented for others and search for issues or bugs. We also reached out to others outside of our team to go through our application.

We also experimented with snapshot testing. This functionality creates a JSON representation of the React components of each page; if the structure or style of any page changes without the snapshot being recreated, it is reported as a test failure. This means that developers will be alerted to any unintended modifications to the user interface.

Unit test definition:

All of our backend unit tests use the JUnit framework. Each one is aimed to test a single specific feature in such a way where every possible input is tested in a repeatable way. We followed the common best-practice standards where all of our unit tests are repeatable, comprehensive, and test a single feature. An example of such a test would be:  
  


System Test Specification

Our application uses comprehensive JUnit tests, as well as a battery of QA tests.

Test Reports per Sprint

Before every demo, we ran our battery of unit tests. We can proudly say that for every single demo, we had fully passing unit tests. Additionally, each Sprint after our test we would have a meeting to discuss bugs and results of our QA testing. We added and closed issues in our GitHub repository based on the finding of our tests.