Software Testing and Debugging

**Milestone 2 - Lessons**

horizontal line

**Introduction**

In this report I will detail the things i gained from developing tests and structure for the Android application MinimalTODO. For an android application that did not have any unit tests there were many challenges surrounding the infrastructure and using several tools for the first time.

**Using Mocks, Fakes, JUnit and Simulating Behavior:**

One of the learnings in this whole project was to understand and utilize the use of mock objects, fake objects and then controlling the behaviour of the mocks to test for all possible cases for failure and testing developer behavior. For example if there is a part of the code where I am handling exceptions, I could ensure that the exception is triggered and handled appropriately.

**Using the Android Framework and Gradle:**

Another positive from this project was understanding how to trigger and build test cases for the entire project. Also, understanding scripting and automation made me realize how powerful test automation can become.

**Coverage alone is not enough:**

Another lesson that was reinforced in my head is that code/test coverage alone does not mean anything and cannot uncover any bugs.

**We cannot test badly written Code:**

I learnt that tests are not a cover for badly written code. In this project I had to change several classes to get them to be test compatible. For example, using dependency injection and construction rather than creating new objects inside methods for example, which can become hard to examine and test.

**Stress Testing:**

Stress testing is the process of determining the ability of a computer, network, program or device to maintain a certain level of effectiveness under any conditions. This process measures the frequency of errors or system crashes. Stress testing benefits by revealing application issues that only become apparent under these extreme conditions. Proper stress tests can also help you uncover the Synchronization and timing bugs, Interlock problems, Priority problems, Resource loss bugs, Memory leaks, Data loss & corruption etc. So, with stress testing you can identify the potential breaking points in your application, which will allow you to correct them before they become expensive issues in production.

For this app, we have done stress testing for 100 Processors and 100 loops. It takes around 115mins to complete entire activity.

*./StressTestApp.sh -NumBackGroundProcesses 10 -NumberLoop 10*

**Monkey Testing:**

The Monkey is a command line tool that runs on your emulator or device and generates random streams of user events such as clicks, touches, or gestures, as well as a number of events. Basically, Monkey is used to stress-test applications that you are developing, in a random yet repeatable manner.

To run monkey testing, we have to launch the Monkey using a command line on our development machine. Because the Monkey runs in the emulator/device environment, we had to launch the application from a shell in that environment and entered Monkey commands directly. Below is the syntax:

$ adb shell monkey [options] <event-count>

and in case of our application the command will look like:

$ adb shell monkey -p com.avjindersinghsekhon.minimaltodo -v 500