# CS 320 Project 2

Andrew Vue

Andrew.vue@snhu.edu

Southern New Hampshire University

**Summary and Reflections Report**  
Your supervisor has asked that you submit a follow-up summary and reflections report to explain how you analyzed various approaches to software testing based on requirements and applied appropriate testing strategies to meet requirements while developing the mobile application for the customer. This report should be based on your experience completing Project One. You must complete the following:

1. **Summary**
   1. Describe your unit testing approach for each of the three features.
      1. To what extent was your approach **aligned to the software requirements**? Support your claims with specific evidence.

The unit testing approach for each of the three features that I took was a slow and steady step. I am new to using unit testing and to be honest I had a difficult time writing the unit tests so it took a lot of research and trial and error because at times I would end up with tests that are not environment independent. I believe I was aligned to the software requirements because I took my time on each test and made sure it was functional before moving on. This may have been inefficient for time, but I believe it helped me learn a lot better than trying to rush things.

* + 1. Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were **effective** based on the coverage percentage?

Initially I was unsure of how effective my JUnit tests were because it was my first time trying out and writing JUnit tests. I tried my best to follow the S.O.L.I.D. class design principle in that I tried to keep my unit tests to one class having one responsibility. I made sure that my code is open for extension, but not modification.

* 1. Describe your experience writing the JUnit tests.
     1. How did you ensure that your code was **technically sound**? Cite specific lines of code from your tests to illustrate.

To ensure that my code was technically sound I tested each of my methods and for different functionality. In this case I initially set up my methods to create a new appointment and assign it a unique ID with parameters, in the test I test for unique IDs and the parameters.

**AppointmentService.java:**

public class AppointmentService {

final private List<Appointment> appointmentList = new ArrayList<>();

private String newUniqueId() {

return UUID.randomUUID().toString().substring(

0, Math.min(toString().length(), 10));

}

public void newAppointment() {

Appointment appt = new Appointment(newUniqueId());

appointmentList.add(appt);

}

**AppointmentServiceTest.java**

@Test

void testNewAppointment() {

AppointmentService service = new AppointmentService();

service.newAppointment();

assertNotNull(service.getAppointmentList().get(0).getAppointmentId());

assertNotNull(service.getAppointmentList().get(0).getAppointmentDate());

assertNotNull(service.getAppointmentList().get(0).getDescription());

service.newAppointment(date);

assertNotNull(service.getAppointmentList().get(1).getAppointmentId());

assertEquals(date,

service.getAppointmentList().get(1).getAppointmentDate());

assertNotNull(service.getAppointmentList().get(1).getDescription());

service.newAppointment(date, description);

assertNotNull(service.getAppointmentList().get(2).getAppointmentId());

assertEquals(date,

service.getAppointmentList().get(2).getAppointmentDate());

assertEquals(description,

service.getAppointmentList().get(2).getDescription());

assertNotEquals(service.getAppointmentList().get(0).getAppointmentId(),

service.getAppointmentList().get(1).getAppointmentId());

assertNotEquals(service.getAppointmentList().get(0).getAppointmentId(),

service.getAppointmentList().get(2).getAppointmentId());

assertNotEquals(service.getAppointmentList().get(1).getAppointmentId(),

service.getAppointmentList().get(2).getAppointmentId());

* + 1. How did you ensure that your code was **efficient**? Cite specific lines of code from your tests to illustrate.

Ensuring efficiency in my test code was difficult because I have never written test code before. I started off by writing tests for my methods using specific scenarios, but as I researched and refactored my code, I learned about designing tests that do not require hard coding tests. As I started to learn JUnit testing, I realized that I didn’t need to hard code specific test cases. This was confusing at first because even now I don’t fully understand JUnit testing. An example would be my test code for the ContactService portion that had a test file like this:

public class Test {

public static void main(String[] args){

Contact contacttest1 =new Contact("10101","AAAbb","zz1zz","1234567891","Hello");

Contact contacttest2 = new Contact("000002222", "ZZEEEEZZ", "BEEEEP","0123456789101112", "GoodGood");

ContactService contactService = new ContactService();

System.out.println(contactService.addContact(contacttest2));

}}

1. **Reflection**
   1. Testing Techniques
      1. What were the **software testing techniques** that you employed in this project? Describe their characteristics using specific details.

To be honest I am not too sure as to what techniques I used were the most efficient, I followed along with the articles on how to write Unit tests and other “best practices” guides, but still struggle with these concepts and techniques. I enjoyed static testing because I think this is the one technique that I’m most comfortable with reviewing my code and making sure I followed the correct syntax structure and avoided silly mistakes. I was very careful with this portion of testing because I know that it would cause me even bigger issues when I try to write unit tests later one –especially since I’m not good at writing unit tests.

* + 1. What are the **other software testing techniques** that you did not use for this project? Describe their characteristics using specific details.

I did not use integration testing because it was not quite necessary and when I tried, I found it difficult to accomplish in time for delivery (assignment due date). I also did not get the chance to try performance testing because the project did not have big loads or stresses.

* + 1. For each of the techniques you discussed, explain the **practical uses and implications** for different software development projects and situations.

The techniques I discussed are good examples of what a thorough testing procedure would include. In most cases unit tests are written by developers before they push their code off to QA so it is important to understand the principles of this. In large corporations and enterprises testing will vary, but usually consists of thorough testing because it is expensive when software breaks after deployment. Stress tests is probably one of the most important if the software being deployed is a large project like a google application used worldwide.

* 1. Mindset
     1. Assess the mindset that you adopted working on this project. In acting as a software tester, to what extent did you employ **caution**? Why was it important to appreciate the complexity and interrelationships of the code you were testing? Provide specific examples to illustrate your claims.

The mindset that I adopted while working on this project was “what can I do to break this”. This is my way of thinking about caution because I wanted to make sure that the code in the project is solid and not break easily. There is also the complexity of writing test code and making sure that I am testing for the correct things. This was the trickiest part of this project. I found it difficult to write tests that would work and then if it didn’t function, I would bounce between if it was my test code that didn’t work or if it was the code that I was trying to test.

* + 1. Assess the ways you tried to limit **bias** in your review of the code. On the software developer side, can you imagine that bias would be a concern if you were responsible for testing your own code? Provide specific examples to illustrate your claims.

Trying to limit bias in my code review is difficult because I as a developer want my code to work. I can see this bias to being an issue because it could cause for tests to not be thorough enough. There are also test cases/scenarios that I may not account for because in my mind as a developer I wrote my code to take contact information, but may not account for different phone numbers or addresses that can cause issues later if I don’t test for it.

* + 1. Finally, evaluate the importance of being **disciplined** in your commitment to quality as a software engineering professional. Why is it important not to cut corners when it comes to writing or testing code? How do you plan to avoid technical debt as a practitioner in the field? Provide specific examples to illustrate your claims.

It is important to not cut corners when writing code because the consequences can be disastrous. In the articles we read about The Explosion of the Ariane 5, this error caused millions of dollars to be lost and could have been avoided with more thorough testing by a third-party company. I think having someone else checking your work is the best way to avoid mistakes and biases. If I ever find myself in a position where I can make these big decisions with big expensive projects like the Ariane 5; I would push for third-party auditors/testers to be added to the project. This redundancy may seem expensive and unnecessary, but I prefer spending the extra for security rather than possibly destroying a whole project that took so much time and money to build.