The JUnit tests performed on each of these classes were to verify that each class works as they should, and that is how I know that the quality is effective. In my appointment class I created a test that creates an appointment, makes a date, gets the description, and then the appointment ID. “@Test

public *void* testCreateAppointment() {

Appointment appointment = appointmentService.createAppointment("01/02/2020", "Meeting with client", "100");

assertEquals("01/02/2020", appointment.getDate());

assertEquals("Meeting with client", appointment.getDescription());

assertEquals("100", apointment.getID());

}

“

This test verifies that evidence of meeting the requirements in this case. I did not enjoy writing JUnit tests at first, but as the semester went on I felt differently. I feel even stronger than I did 7 weeks ago that testing is incredibly important in programming. I believe that everything should be periodically tested throughout development, and everyday as changes are made. I made sure my code was technically sound by following many different YouTube videos, and creating clean code. To verify this I will use the next lines of code following the previous shared code.

@Test

public *void* testDeleteAppointment(){

Appointment appointment = appointmentService.createAppointment("01/02/2020", "Meeting with client", "100");

appointmentService.deleteAppointment(appointment);

assertEquals(AppointmentStatus.DELETED, appointment.getStatus());

}

This test verifies that not only can you make the appointment correctly, but delete the appointment as well. Each class has a test verifying that each requirement is correct. It is efficient because I used clear code and strings to show indication of what I was programming. As well as, strings with effective clear names.

I performed unit tests on each of the different classes to make sure they were aligned to the software requirements. I used unit testing to verify the individual components of the software, I used system testing to test the entire application, and performance testing to ensure it was running aesthetically pleasing, while also properly. I made an appointment class to make appointments using the mobile app, the appointments can be created, deleted, has a date, description, and ID. I made a contact class to add contacts into the app, these contacts can be added, or deleted and tasks to add, update or delete.

Another software testing technique not used in this project was security testing. This tests the vulnerabilities in the software to meet security requirements in the software development life cycle. This is an important test especially for personal information, for example an implication could be if you are at a medical setting, the cybersecurity needs to be airtight for HIPAA regulations. That is not only your client, but all those peoples information that needs to be protected. This fact alone makes this test extremely important.

As a software tester in this project, I saw a different side of programming, a side I had not seen before. I learned how important it truly is to run tests on your code, and write commented code. I employed caution with each test I ran against the class, by making sure the test was the best way to write it. I researched each test, and made sure each string was written clearly. It is important to appreciate the complexity of the code your testing, because not all code is the same, and when coding a bigger software it is important to write tests for each of the commits you do. Everyone is taught the same, to write it the same, but everyone does it a little bit differently. That is why we use comments in code. Learning about testing is primitive to writing code to program software, because without testing how do you know it works?

I never thought of having bias in writing code, I write to the requirements, and make sure its clearly written. I would argue a defense that writing your own tests to test your own code, only matters if it’s wrong and the software doesn’t come out effective. I would argue that writing tests within the code, helps for a more efficient outcome for the final project, because you tested it all along rather than all at once at the end, playing a guessing game to see how much is wrong.

I have probably said this too many times, but it is SO SO SO important to test code. I found that when working within a big tech company, they are always trying to cut corners and us as the developers, need to verify. It is your code on the line, you need to make sure as an individual that it works the best of your ability. In school they test your knowledge, so test the softwares knowledge. Ask the question, did you write code to tell it do what you need it to do? Not only that, errors, mistakes happen all the time, and tests find those right away. It is easier to find a semi colon, or a miss written appointment class in 20 lines then 2000.