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.

My entire approach was based on the software requirements. I find that you can be a lot more successful in creating a project that your client wants by having a clear understanding of what they are looking for in their software. As I completed each class, I made sure to also write in a test case for each requirement in order to prove that it was included and functions properly.

* + 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?

The quality of my Junit tests is very high. My Junit tests confirmed that the requirements were met and would give any client looking over the report confidence that their features are included and working properly. I believe that my coverage was close to 100%. Every line was covered and tested to ensure that it not only met the requirements but functioned alongside the other requirements in a way that was efficient.

* 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.

I ensured that my code was technically sound in each of the tests. Each test would try to do various things within the classes such as test the add feature of the Task or the update feature of the contact service. Something that I have found is that it is always the error handling that can make a difference between good and bad code.

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

I ensured that the code was efficient by making sure there was no spaghetti code to solve the various requirements. Each of the requirements got its own function or variable depending upon the nature of the requirement. Unnecessary steps can lead to a slower program and that is not fun for anyone.

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

Throughout this project, I used static testing techniques of reading over my code to make sure that it will do what I want it to. I also made sure to use dynamic testing through the Junit tests. I used exploratory testing as I went through the code to ensure everything was in order.

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

An example of a testing technique that I did not use was checklist testing. Checklist testing is when you gather a list of issues and go through and test the one by one.

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

I believe that checklist testing would be great for a situation where you are actively receiving feedback from users and can make a checklist. Exploratory testing is more broadly used. In just about all coding scenario developers use Exploratory testing to a degree.

* 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.

As I was acting as a software tester, I tried to employ caution by thinking of ways that a user might accidently break the code. It is important to appreciate the complexity of the code because if you do not you may miss something that a user might do. By thinking about all the ways that a user might interact with the code you can keep the number of errors to a minimum.

* + 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.

It can be hard to be both the developer and the tester. I did not think about bias much, but I can see how it could be easy to make a test pass. As a developer you know how the code is supposed to interact and can implement the test in such a way that you always receive the desired result.

* + 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 stay disciplined in providing quality solutions as a software engineer. By cutting corners you can run into issues such as broken code, missing requirements, or a lack of upgradability. As a software engineer, I will always try to have clarity in my projects. I will know what it is that is being asked of me, and create programs that are efficient and tested properly for accuracy.