David Greene

CS-320 Journal Assignment

I find it easier to directly answer the questions posed rather than trying to turn this into an essay:

**1. To what extent was your testing approach aligned to the software requirements?**

My testing approach appears to be aligned with the software requirements. My tests cover various scenarios and validate the expected behavior of the services, such as adding, updating, deleting, and retrieving contacts and tasks. This demonstrates a good approach to testing that aligns with the requirements.

**2. Defend the overall quality of your JUnit tests for the contact service and task service.**

The overall quality of my JUnit tests for the contact service and task service seems to be good. My tests are well-structured and cover different aspects of the services' functionality. They use assertions to validate the expected results, ensuring that the services behave correctly. Additionally, the tests include scenarios that test both valid and invalid inputs, which helps ensure the robustness of the services.

**3. How did you ensure that your code was technically sound? Cite specific lines of code from your tests to illustrate.**

To ensure technical soundness, it's important to follow best practices and adhere to coding standards. In my TaskServiceTest class, the following lines demonstrate technical soundness:

assertTrue(taskService.getTasks().containsKey("1234567890"));

assertEquals(task, taskService.getTasks().get("1234567890"));

These lines validate that a task with a unique ID is added to the task service and can be retrieved correctly. The assertions ensure that the expected results match the actual results, which is a good practice for ensuring technical soundness.

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

Efficiency in code can be achieved by optimizing algorithms, minimizing unnecessary operations, and utilizing appropriate data structures. I do feel I have used the appropriate data structures; not sure I have optimized any algorithms however. Maybe in the future.

However, it's important to note that the efficiency of the code can be improved by considering factors such as time complexity and memory usage. For example, in the TaskServiceTest class, I think the following lines could have beeen improved:

assertTrue(taskService.getTasks().containsKey("1234567890"));

assertEquals(task, taskService.getTasks().get("1234567890"));

Instead of using the containsKey and get methods, it might have been more efficient to directly access the task using the ID. This would avoid unnecessary method calls and improve efficiency.