Project 2

As a software engineer at Grand Strand Systems, I recently completed the development and testing of three key services (contact, task, and appointment). This project focused on implementing these services while ensuring functionality through unit testing with JUnit. In this report, I will summarize my unit testing approach for each feature, discuss the overall quality of my tests, and reflect on the testing techniques employed.

For the contact service, my unit testing approach centered around validating the core functionality of adding, updating, and deleting contacts. The primary goal was to ensure that the contact objects followed to the project’s requirements. This includes names, valid phone numbers, and unique IDs. Each unit test targeted a specific function to verify whether it behaved as expected. For instance, the test case testAddContact() ensured that contacts with correct input were successfully added, while testAddContactNotFound() checked for failures when inputs like name or phone number were invalid.

The task service required thorough testing to confirm that tasks were being managed correctly, including the ability to add, update, and delete tasks. It required tasks to have a unique ID and a description. For example, testAddTask() verified that tasks with valid data could be added, while testUpdateTask() ensured that only tasks with appropriate data could be updated.

The appointment service involved not only managing appointment objects but also validating date formats and length restrictions on appointment descriptions. In this case, tests like testAddAppointment() checked whether an appointment with a properly formatted date could be added, while tests like testInvalidApointmentDate() ensured the system responded correctly to date format violations.

Overall, my unit testing approach for each service was aligned with the software requirements. The tests were structured to check the critical requirements of each feature, ensuring that the services met their functional specifications.

To assess the quality of my JUnit tests, I relied on the principle of code coverage to measure how much of the codebase was exercised by the tests. Achieving high code coverage was essential to ensure that each significant path in the code was verified.

One of the key factors in maintaining high test quality was ensuring that each test covered the essential branches and conditions of the code. For example, the test case testDeleteContactSuccess() ensured that a contact was deleted when a valid ID was provided. This helped confirm that the system responded correctly in all tested scenarios, adding to the overall quality of the tests.

Writing JUnit tests provided an opportunity to practice creating technically sound and efficient code. This test ensured that invalid phone numbers were rejected by the system, fulfilling the functional requirement of only accepting properly formatted numbers.

To ensure efficiency, I made use of parameter tests to reduce redundancy. By minimizing repetition, I ensured that my code was more maintained and ran efficient.

The primary software testing technique employed in this project was unit testing, which focuses on testing individual components of the application in isolation. Unit tests allow for validation of functionality, ensuring that each method behaves as expected. This ensured that the application met the end-user requirements.

Although I relied heavily on unit testing, there are other techniques I did not use in this project, such as integration testing and regression testing. Integration testing would have been useful for verifying that the contact, task, and appointment services worked together seamlessly, ensuring that changes in one service didn’t break functionality in another. Regression testing, on the other hand, would have been beneficial for automatically checking if changes introduced new bugs. In future projects, I plan to incorporate these techniques to enhance the reliability of the software.

It was crucial to appreciate that changes in one service could have unforeseen effects on other services. For example, while testing the appointment service, I paid attention to how changes in the date validation logic could impact related features like task scheduling.

Finally, as a software engineering professional, I recognize the importance of maintaining a high standard of quality. Cutting corners in writing or testing code can lead to technical debt, which often results in long-term issues that are costly to fix. In this project, I was careful to test and ensure proper error handling to avoid future problems. Moving forward, I will continue to emphasize code quality by using tools like static analysis and peer reviews, ensuring that I maintain a disciplined approach to software development.

In conclusion, this project reinforced the importance of a comprehensive and disciplined approach to software testing. By focusing on unit testing, I ensured that the contact, task, and appointment services were functional. Moving forward, I plan to incorporate additional testing techniques, such as integration and regression testing, to further improve the quality of the software I develop.