Reflecting on the development of Project One, my approach to unit testing for the Contact, Task, and Appointment services was methodical and aligned closely with the software requirements. Each service required meticulous testing to ensure all aspects functioned as intended.

For the Contact Service, my focus was on validating each field of a contact. For instance, in `ContactTest.java`, tests like `testContactIdNull` and `testContactIdExactLength` were crucial to ensure the contact ID met its unique and length-specific criteria. Similar detailed tests were conducted for other fields like first name, last name, phone number, and address. This approach was perfectly aligned with the requirements, which demanded strict adherence to field-specific constraints. A screen shot of a computer program

Description automatically generated

In the Task Service, tests in `TaskTest.java` such as `testNameValidation`A computer screen with text

Description automatically generated and `testDescriptionValidation`A screen shot of a computer program

Description automatically generated ensured that task names and descriptions adhered to their respective length limitations. Additionally, in `TaskServiceTest.java`, tests like `testAddTask` and `testDeleteTask` validated the service's ability to manage tasks effectively, demonstrating alignment with the functional requirements.

The Appointment Service required a different testing strategy, especially for date validations. In `AppointmentTest.java`, tests like `testAppointmentCreationValid` and `testAppointmentCreationPastDate ensured that appointments were only created with future dates, in line with the project requirements.

The overall quality of my JUnit tests can be defended based on the coverage percentage. For example, in `ContactTest.java`, the tests covered a wide range of scenarios, from null values to maximum length validations, ensuring a high coverage percentage. This comprehensive testing approach was mirrored in the other services as well, confirming that most, if not all, code paths and scenarios were tested.

Writing these JUnit tests was a learning experience that deepened my understanding of the nuances in each service. It required a balance between rigorous testing and efficient code implementation. For instance, in `AppointmentServiceTest.java`, the use of setup methods and assert statements ensured that tests were not only thorough but also concise.

To ensure technical soundness, I relied on assert statements for accuracy and exception handling for robustness. For instance, `assertThrows` in `ContactTest.java`A screen shot of text

Description automatically generated ensured that invalid inputs were correctly handled. Similarly, efficient code patterns, like reusing setup methods in `TaskServiceTest.java`, made the tests not only effective but also efficient.

Reflecting on the software testing techniques employed, unit testing was the primary method used. It allowed for testing individual components in isolation, ensuring each part functioned correctly before integrating them into the larger application. However, other techniques like integration testing, which tests the interactions between modules, and performance testing, which measures the application's performance under load, were not used in this project but are crucial in more complex, integrated projects.

My mindset while working on this project was one of caution and diligence. I appreciated the complexity and interrelationships of the code I was testing. For example, understanding how the update methods in `ContactService.java` impacted the overall functionality of the service required a cautious approach. Limiting bias in code review was vital, as testing one's own code could lead to overlooking flaws. I approached this by rigorously questioning my assumptions and testing for edge cases.

Lastly, my commitment to quality was unwavering. Cutting corners in writing or testing code can lead to technical debt and unreliable software. I plan to continue this disciplined approach in my future work, prioritizing thorough testing and review to build robust and efficient software solutions.