Lee Kitchen

CS-320

18FEB2024

7-2 Project Two

**Summary**

In developing the mobile application for the client, rigorous unit testing was conducted to ensure compliance with the specified requirements for the ContactService, TaskService, and AppointmentService components. Each feature underwent a meticulous unit testing approach tailored to its unique functionalities. For the ContactService, tests were designed to validate the addition, deletion, and updating of contacts, ensuring adherence to requirements such as unique contact IDs and mandatory fields. Similarly, the TaskService and AppointmentService were subjected to comprehensive unit tests covering the creation, deletion, and modification of tasks and appointments, respectively. These tests were directly aligned with the software requirements, with each test method specifically targeting a particular requirement. The quality of the JUnit tests was upheld through rigorous coverage monitoring, maintaining an 80% coverage threshold to ensure thorough examination of critical code paths. From the following screenshot of my coverage. You can see I achieved this. A screenshot of a computer program

Description automatically generatedI did experience some issues with how it was reporting for the ContactTest, all of the coverage for more tests hit more than the minimum of the 80%. The writing of these JUnit tests proved to be an enlightening experience, demanding careful consideration of edge cases and boundary conditions to achieve comprehensive test coverage. The resulting tests were not only technically sound but also efficient, exemplified by specific lines of code validating functionality within the ContactService.A screen shot of a computer program

Description automatically generated

As seen in the provided screenshot, the JUnit test combines both the null ID and length greater than 10 in one test rather than doing it in two.

**Reflection**

Throughout the project, the unit testing techniques that were employed ensured the correctness and robustness of the application components. While these techniques proved effective in ensuring the functionality of individual units and their interactions, other techniques such as regression testing and acceptance testing were not utilized but could offer benefits in future development iterations. Regression testing would serve to maintain code stability amidst frequent changes, while acceptance testing would validate that the software meets customer expectations. Adopting a cautious mindset during testing was imperative, necessitating an appreciation of the complexity and interrelationships within the codebase to identify potential pitfalls and edge cases. This mindset was particularly crucial when considering scenarios such as overlapping appointments within the AppointmentService. Furthermore, efforts were made to limit bias in code review by approaching the evaluation objectively, focusing on defect identification rather than confirmation of preconceived notions. It was acknowledged that bias could pose a concern if responsible for testing one's own code as a developer, potentially leading to oversight of critical issues or assumptions regarding implementation correctness. Maintaining discipline and commitment to quality emerged as fundamental principles in software engineering practice. Cutting corners in code writing or testing was recognized as detrimental, as it could result in the accumulation of technical debt, hindering long-term maintainability and scalability. For instance, rushing the implementation of the AppointmentService without thorough testing could lead to unforeseen bugs or violations of requirements, necessitating increased maintenance efforts in the future. By investing time upfront to write comprehensive tests and code, technical debt could be mitigated, ensuring the delivery of a robust and reliable application. Ultimately, the importance of adhering to best practices and avoiding shortcuts was underscored, serving as a blueprint for navigating future software engineering endeavors with diligence and integrity.

References

Boni Garcia. (2017). Mastering Software Testing with JUnit 5 : A Comprehensive, Hands-on Guide on Unit Testing Framework for Java Programming Language. Packt Publishing.

Hambling, B., Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (2019). *Software testing : An istqb-bcs certified tester foundation guide - 4th edition*. BCS Learning & Development Limited.