Darkenley Minviel

CS 320

Dr. Cross

**Project two**

Testing is a very important part of software development. My approach consisted of testing as early as possible during the coding writing step. The initial testing was a static testing of the code, followed by many manual corrections then I listed the software requirements as bullet point to which I created an attached test.

The coverage tools within Eclipse can be used identify what areas of the project need more attention. My Junit test cases are effective as they were able to cover about 81% of the written code and generated the desired outcomes. For example, in Contact and Task service, one of the requirements is that the Contact or Task ID should not be longer than 10 digits and should not be a null value. To verify the effectiveness of the code, I have written two tests one to test for longer ID and one for a null value.

Software testing can be split into small components called units; each unit test for a small portion of code, as this can help you find bugs in software and typing or logical errors. Our program has gone through static and dynamic testing to ensure soundness. Overall, we have ensured that the quality of the code is clear, simple, well tested, bug-free. The attached pictures show how requirements were handled and comments to guide any future editors of the codes and the next picture reflects the testing of specific requirements.

Text

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

My code was efficient since it was able to pass all the Junit test created, on top of that it runs without producing any errors. The logic was right and not off. And as it is reflected in the picture above, invalid entry or entry that did not meet the requirements was handled by throwing an illegal argument exception. All the different requirements were tested for.

Additionally, coding requires a lot of focus, discipline, and structure. Anyone can code however to be a good developer or programmer you need the quality and principles mentioned earlier. An essential part of software development is software testing. There are many of them and they are split into 2 main categories, the functional and non-functional testing. Function testing include unit testing, integration testing, system testing, and acceptance whereas non-functional testing include performance, security, usability, and compatibility testing. In the completed milestones, I have made used of unit testing and acceptance testing. Our unit testing involves testing every one of our functions for the requirements set for the milestones. Our acceptance involves making sure that all our unit testing was passed and that the classes were checked for logical errors and run for static error.

In the milestones, we did not make any use of integration testing, system testing (functional), and neither of performance, security, usability, and compatibility testing. The use of these testing is necessary when working on a full-scale program. Since our milestone was basically about couple requirements about input, there were not necessary. It is good to note that we could have conducted performance testing if part of the requirement involved speed and processing of a large amount of data at once.

The unit testing allows to split the requirements into small portions and test them. It is highly recommended instead of testing the system as a whole because it helps prevent mistake that could be missed in dynamic testing. Integration testing allows to check how 2 different modules interact with each other for example testing the login component of a software while testing how it interacts with a basket in a store mobile app. System testing follows integration testing, it is testing the system as a whole for compliance with the requirements. Security testing is crucial when dealing with personal data and required when dealing with financial data. Performance testing is about quality. If your software fails to pass performance testing, then you are more likely to have a slow app that no consumer would want to use. Usability is a key testing before release because it gives you firsthand feedback of a potential user on your system. And compability allows to test that your program will indeed work on your targeted devices or browser.

In developing this project, I use the safety-critical system approach. According to Osepchuk (2021), “Safety-critical systems are those systems whose failure could result in loss of life, significant property damage, or damage to the environment”. Even though this project does not involve life or death situation but it is good to keep practice so it can become an habit. In the future when developing, the loss might not be a life, but it might vast enough to create a world crisis of data loss. It was important to appreciate the complexity and interrelationship of the code because they were created separately and function as they own entity to later be combine and become one system.

When reviewing our work our own work as a developer, we tend to not focus on the possible negative because we hope that it is working and are more focus on the positive result. “Code tends to be less than it could be when only one person looks at it. It’s often more complex than necessary (Miller, 2018). It’s often buggy. It’s often shortsighted. It may seem perfectly clear to the person who wrote it, but not so clear for the first person that reads it.” A professional should never be tempted in cutting corners because it can be very costly at the end. Software security is important, and crucial depending on the affected sector. Cutting edges in software development designed for hospitals might cause your imprisonment for failure of HIPAA.

**Reference**

Osepchuk, Blaine (2021). Safety-Critical Software: 15 things every developer should know. Retrieved on April 12, 2022 from https://smallbusinessprogramming.com/safety-critical-software-15-things-every-developer-should-know/

Miller, Roy (2018). Coding with Two Sets of Eyes (Part 1). Retrieved on April 12, 2022 from https://blog.rolemodelsoftware.com/coding-with-two-sets-of-eyes-part-1-82b1e3c14a83