Project 2

Matthew Courts

Feb 25, 2024

In my approach to testing the software, I tested that the requirement conditions were met as described in the requirements sections. Test conditions were isolated to the matching class and did not test for overlapping failures from other classes. This containerizes the requirements for each class or unit individually. For each unit, the requirements listed in the prompt specified the customer's needs or requests. This is the goal of the produced code to match the customer needs, with quality analysis the JUnit testing allows us to test specific use cases specifically. In this case, expected pass and fail conditions were tested for each unit. In terms of isolation of tests, the requirements for each unit were only held to each unit test.

The code is structured with data validation handled at the beginning of a function with the actual code being performed after the data has been validated removing excess code operations if the input validation fails. Code efficiency is performed by using internal tools for data management, the List<> data type allows for the management of custom types with built-in feedback on successes or failures when managing the items within lists. The ability to reference list items and call internal class functions on the item removes the need to capture temporary data creating memory overhead and increasing performance.

/\* Contact Class, set address property \*/

public void set\_address(String \_address) {

/\* Exit conditions\*/

if (\_address == null) {

throw new IllegalArgumentException();

}

if (\_address.length() > 30) {

throw new IllegalArgumentException();

}

/\* set property \*/

this.\_address = \_address;

}

/\* Contact Service , ContactList during constructor call \*/

private final List<Contact> ContactList;

public ContactService() {

this.ContactList = new ArrayList<>();

}

For the Grand Strand Systems appointment service, three units were written to handle the contact, task, and appointment data that would be captured. To ensure code quality, and function the units are tested along with how they interact with each other to ensure that the code will meet the requirements set by Grand Strand Systems. To meet these requirements manual and automatic tests were performed and evaluated. Manual code tests occur during the development of the units but will continue into the development of the automated tests. With the introduction of automatic testing, we can ensure that each requirement is met and find any other issues during manual testing. Since Java was employed, JUnit4 was used to create tests for each unit. These tests would look for not only successes but also failures and null conditions ensuring the units are handling data validation. With JUnit4 unit-level testing on each function and in many cases testing that some functions were properly restricted, for instance with managing the unique IDs.

While testing was handled for the units individually there are other tests to be performed later in the development of the overall application. Integration testing would be the next major step testing and validating that each unit can interact with each other. This can be a major hurdle to cross early on as integration begins and unexpected problems are encountered. This can also lead to the assessment of Functional tests on the units to ensure they meet the business requirements of the customer. This allows for an early and low-level measurement of how the app meets the needs of the business. Once a working version of the app is in operation End-to-end tests will be employed to test user actions and behavior. This is great for testing user actions along with call-to-actions ensuring their behavior meets expectations.

Acceptance Testing allows for a more formal approach to testing the software and requires a fully functioning application focusing on the business requirements of the application ensuring they are met. This will test user actions from start to finish of the application and can be a method of automated and manual tests. Performance tests provide another look into the application regarding how well it runs and handles traffic loads. In terms of web apps, the application should be scalable along with handling a certain amount of traffic at once. This can also look closer at the performance of individual units often profilers are put to use with the measurement of time and number of calls to a specific feature. This can be very extensive and time-consuming testing but creates focus on not only the application is affordable to the company but also easily usable to the end users.

My mindset working on this project is focused on quality control through testing. This requires caution and awareness of the structure of the code, its logic, and foundation libraries. Being in the right mindset also requires understanding the coding language chosen as misusing the programming language in manners it was not intended which can lead to quality issues along with expanding on the code later on. Static tests and code reviews can be applied during development to ensure quality while being designed and coded, this will be further expanded into unit tests with further testing being applied automatically or manually with later development. With Static testing Code reviews, walkthroughs, and inspections are used to identify any defects along with validating or improving software quality.

Bias can be a major concern with development as this can vary with individual developer experiences. With code bias, code reviews can stand out strongly by validating the code structure and methodology.

*Cognitive biases are a result of numerous factors and depend on how individuals–including software developers–think. Some biases occur due to our limited cognitive capacity (e.g. Availability bias may cause developers to choose solutions based on examples that are readily available in memory), while some are a by-product of an individual’s development style.* (Oregon State University, 2020)

Code styles are another way to control developer bias by creating a standard on how to code in specific conditions or how to apply a structure. This can be taken further since a global code style in an organization leads to common patterns with naming, structure, and formatting. This leads to consistency between individual developers allowing easier access to code reviewing and development.

Humans prefer to read columns—think about effective newspaper and website layouts. Additionally, remember that the interesting part of a statement happens on the right end. So if a statement runs off the right edge of the screen, it’s easy to miss the importance of what’s going on, either because you form judgments too quickly or get lost endlessly scrolling right. (Chien & Clemons, na)

Quality begins during the development phase of any code, setting developers along a course that is maintained with code reviews and other tests. All this is good but a disciplined developer will understand the goals of the project and how to apply them to the code. Being able to see the operation of the code you are working on from the eyes of the end user is very critical. Not understanding this or cutting corners will lead to unexpected software behaviors that can lead to damage, downtime, or even loss of life. In my daily work, I write code that will produce machine code used in industrial cutting operations. Not testing, or understanding the goals and needs of the customer will lead to damage and downtime. There even be some rare cases where an operator could be injured by failure in the coding part. Testing is critical, validating that the change does not affect other parts of the code, and validating I do not introduce any new behaviors in the code operation. There does become a fine line of how far to test any code changes but development can be too long when incorporating endless testing. This is where experience comes with the underlying libraries and data I use daily. Having a common coding style and structure allows for rapid development of a new feature but also for debugging and testing.